

```
1  HOL : Spring MVC
2  -----
3  Task1. Spring MVC Demo
4  1. Package Explorer > right-click > New > Spring Legacy Project
5  2. Select Spring MVC Project
6  3. Project name : HelloWorldWeb
7  4. Next
8  5. Enter a topLevelPackage : com.example.biz
9  6. Finish
10
11
12 7. HelloWorldWeb project 수정하기
13 1)HelloWorldWeb > right-click > Properties
14 2)Java Compiler > JDK Compliance > Compiler compliance level : 13 으로 설정
15 3)Apply click
16 4)Build the project now? > Yes click
17 5)Java Build Path > Libraries tab > Edit click > Select [Workspace default JRE (jdk-13.0.2)]
18 > Finish
19 6)Apply click
20 7)Project Facets > Java > 13
21 8)Runtime tab > Check [Apache Tomcat v9.0]
22 9)Apply and Close click
23 10)Build the project now? > Yes click
24
25 8. pom.xml 수정하기
26 <properties>
27     <java-version>13</java-version>
28     <org.springframework-version>5.2.5.RELEASE</org.springframework-version>
29     <org.aspectj-version>1.9.5</org.aspectj-version>
30     <org.slf4j-version>1.7.30</org.slf4j-version>
31 </properties>
32 ...
33 <dependency>
34     <groupId>log4j</groupId>
35     <artifactId>log4j</artifactId>
36     <version>1.2.17</version>
37 ...
38 <dependency>
39     <groupId>javax.servlet</groupId>
40     <artifactId>javax.servlet-api</artifactId>
41     <version>4.0.1</version>
42     <scope>provided</scope>
43 </dependency>
44 <dependency>
45     <groupId>javax.servlet.jsp</groupId>
46     <artifactId>javax.servlet.jsp-api</artifactId>
47     <version>2.3.3</version>
48     <scope>provided</scope>
49 </dependency>
50 <dependency>
51     <groupId>org.junit.jupiter</groupId>
52     <artifactId>junit-jupiter-api</artifactId>
53     <version>5.6.2</version>
54     <scope>test</scope>
55 </dependency>
56
57
```

58 9. pom.xml > right-click > Run As > Maven install
59 [INFO] BUILD SUCCESS
60
61
62 10. Servers View 추가하기
63 1) Window > Show View > Other > Server > Servers
64 2) Open click
65
66
67 11. Tomcat Server 추가하기
68 1) In Servers View, [No servers are available. Click this link to create a new server...] click
69 2) Apache > Tomcat v9.0 Server Select
70 3) Next
71 4) Name : Apache Tomcat v9.0
72 5) Tomcat installation directory : C:\Program Files\apache-tomcat-9.0.33
73 6) JRE : jdk-13.0.2
74 7) Finish
75 8) HelloWorldWeb project > right-click > Properties > Project Facets > Select Java >
Change Version 13
76 9) Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close
77
78
79 12. HelloWorldWeb Project right-click > Run As > Run on Server > Finish
80
81
82 13. <http://localhost:8080/biz/>
83
84 Hello world!
85
86 The time on the server is April 19, 2020 at 6:53:00 PM KST.<--원래 한글 깨짐
87
88
89 14. 한글 깨짐을 수정하려면 src/main/webapp/WEB-INF/views/home.jsp에서
90 <%@ page session="false" pageEncoding="UTF-8" contentType="text/html;
charset=UTF-8"%>로 수정
91
92
93 15. Context name 변경하기
94 1) Package Explorer에서 Servers/Tomcat v9.0 Server at localhost-config/server.xml에서 다음과
같이 수정한다.
95 -path="/biz" --> path="/demo"
96 <Context docBase="HelloWorldWeb" path="/demo" reloadable="true"
source="org.eclipse.jst.jee.server:HelloWorldWeb"/>
97
98 2) 수정 후 restart 하면 <http://localhost:8080/biz> --> <http://localhost:8080/demo> 로 변경됨
99
100
101
102 -----
103 Task2. resources Folder 이용하기
104 1. Image 경로 알아내기
105 1) src/main/webapp/resources/에 images Folder를 STS Package Explorer에서 생성한다.
106 2) Internet에서 적당한 image를 src/main/webapp/resources/images/에 다운로드한다.
107 3) home.jsp에 아래 code를 추가한다.
108 <p></p>
109 4) HelloWorldWeb Project right-click > Run As > Run on Server > Finish
110 Image가 잘 나온다.
111

```

112
113 2. Image 경로 변경
114 1)apple.jpg image 경로를 src/main/webapp/images/로 이동.
115 2)하지만 이렇게 하면 image가 보이지 않는다.
116 3)왜냐하면, servlet-context.xml에서 resource의 경로는 <resources mapping="/resources/**"
    location="/resources/" />이기 때문.
117 4)즉, 기본적으로 resources folder 아래에서 resource를 찾는다.
118
119
120 3. <resources />추가
121 1)다시 resources Folder 하위로 images Folder를 이동
122 2)/src/main/webapp/하위에 images Folder를 생성하고 image를 넣고 home.jsp에 아래의 code를 추가한
    다.
123 <p></p>
124 <p></p>
125 3)실행하면 아래부분의 image는 보이지 않는다.
126 4)왜냐하면 새로 추가한 images Folder는 servlet-context.xml에서 설정하지 않았기 때문.
127 5)Image를 보이게 하기 위해 servlet-context.xml에 아래의 Code를 추가한다.
128 <resources mapping="/resources/**" location="/resources/" />
129 <resources mapping="/images/**" location="/images/" />
130 6)Project right-click > Run As > Run on Server > Restart >
131 -image 2개가 제대로 보인다.
132
133
134 -----
135 Task3. Controller Class 제작하기
136 1. 제작순서
137 1)@Controller를 이용한 class 생성
138 2)@RequestMapping을 이용한 요청 경로 지정
139 3)요청 처리 method 구현
140 4)View 이름 return
141
142 5)UserController class 생성
143 -src/main/java/com.example.biz > right-click > New > Class
144 -Name : UserController
145 -Finish
146
147 package com.example.biz;
148
149 import org.springframework.stereotype.Controller;
150
151 @Controller
152 public class UserController {
153
154 }
155
156
157 2. 요청 처리 method 생성
158
159 package com.example.biz;
160
161 import org.springframework.stereotype.Controller;
162 import org.springframework.ui.Model;
163 import org.springframework.web.bind.annotation.RequestMapping;
164 import org.springframework.web.bind.annotation.RequestMethod;
165 import org.springframework.web.servlet.ModelAndView;
166
167 @Controller

```

```

168 public class UserController {
169     @RequestMapping("/view")
170     public String view(Model model){
171         model.addAttribute("currentDate", new java.util.Date());
172         return "view";
173     }
174
175     @RequestMapping("/user")
176     public String user(Model model){
177         model.addAttribute("username", "한지민");
178         model.addAttribute("userage", 24);
179         model.addAttribute("job", "Developer");
180         return "user";
181     }
182
183     @RequestMapping("/fruits")
184     public String fruits(Model model){
185         String [] array = {"Apple", "Mango", "Lemon", "Grape"};
186         model.addAttribute("fruits", array);
187         return "fruits";
188     }
189 }

```

3. View에 Data 전달

1)view.jsp 생성

```

194 -src/main/webapp/WEB-INF/views > right-click > New > Other > Web > JSP File
195 -Next
196 -File name : view.jsp
197 -Finish

```

```

199 <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
200 <!DOCTYPE html>
201 <html>
202     <head>
203         <meta charset="UTF-8">
204         <title>Insert title here</title>
205     </head>
206     <body>
207         <h1>view.jsp 입니다.</h1>
208         현재 날짜와 시간은 ${currentDate} 입니다.
209     </body>
210 </html>

```

2)fruits.jsp 생성

```

213 -src/main/webapp/WEB-INF/views > right-click > New > Other > Web > JSP File
214 -Next
215 -File name : fruits.jsp
216 -Finish

```

```

218 <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
219 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
220 <!DOCTYPE html>
221 <html>
222     <head>
223         <meta charset="UTF-8">

```

```

224     <title>Insert title here</title>
225     </head>
226     <body>
227         <h2>fruits.jsp</h2>
228         <ul>과일 종류
229         <c:forEach items="${fruits}" var="fruit">
230             <li>${fruit}</li>
231         </c:forEach>
232         </ul>
233     </body>
234 </html>

```

3)user.jsp 생성

```

237 -src/main/webapp/WEB-INF/views > right-click > New > Other > Web > JSP File
238 -Next
239 -File name : user.jsp
240 -Finish

```

```

241
242     <%@ page language="java" contentType="text/html; charset=UTF-8"
243         pageEncoding="UTF-8"%>
244     <!DOCTYPE html>
245     <html>
246     <head>
247     <meta charset="UTF-8">
248     <title>Insert title here</title>
249     </head>
250     <body>
251         <h2>user.jsp</h2>
252         <ul>
253             <li>Name : ${username}</li>
254             <li>Age : ${userage}</li>
255             <li>Job : ${job}</li>
256         </ul>
257     </body>
258 </html>

```

260 4) <http://localhost:8080/demo/view>

261 5) <http://localhost:8080/demo/fruits>

262 6) <http://localhost:8080/demo/user>

265 4. View에 ModelAndView 객체로 data 전달

266 1)UserController.java에 아래의 코드 추가

```

267
268     @RequestMapping(value = "/demo", method = RequestMethod.GET)
269     public ModelAndView demo() {
270         /*
271         ModelAndView mav = new ModelAndView("view2");
272         mav.addObject("username", "한지민");
273         mav.addObject("currentDate", new java.util.Date());
274         return mav;
275         */
276         ModelAndView mav = new ModelAndView();
277         mav.addObject("userid", "example");
278         mav.addObject("passwd", "12345678");
279         mav.setViewName("/demo");
280         return mav;
281     }

```

```
282
283 2)src/main/webapp/WEB-INF/views/demo.jsp 생성
284
285 <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
286 <!DOCTYPE html>
287 <html>
288     <head>
289         <meta charset="UTF-8">
290         <title>Insert title here</title>
291     </head>
292     <body>
293         아이디 : ${userid} <br />
294         패스워드 : ${passwd}
295     </body>
296 </html>
297
298 3)<a href="http://localhost:8080/demo/demo">http://localhost:8080/demo/demo
299     아이디 : example
300     패스워드 : 12345678
301
302
303 5. Controller class에 @RequestMapping 적용
304 1)src/main/java/com.example.biz.StudentController.java 생성
305
306 package com.example.biz;
307
308 import org.springframework.stereotype.Controller;
309 import org.springframework.web.bind.annotation.RequestMapping;
310 import org.springframework.web.bind.annotation.RequestMethod;
311 import org.springframework.web.servlet.ModelAndView;
312
313 @Controller
314 @RequestMapping("/bbs")
315 public class StudentController {
316
317     @RequestMapping(value="/get", method = RequestMethod.GET)
318     public ModelAndView getStudent() {
319
320         ModelAndView mav = new ModelAndView();
321         mav.setViewName("/bbs/get"); // /WEB-INF/views/bbs/get.jsp
322         mav.addObject("name", "한지민");
323         mav.addObject("age", 25);
324         return mav;
325     }
326 }
327
328 2)src/main/webapp/WEB-INF/views/bbs/get.jsp
329 -views > right-click > New > Folder
330 -Folder name : bbs
331 -Finish
332
333 <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
334 <!DOCTYPE html>
335 <html>
336 <head>
337 <meta charset="UTF-8">
```

```
338     <title>Insert title here</title>
339     </head>
340     <body>
341         학생 이름 : ${name} <br />
342         학생 나이 : ${age}
343     </body>
344 </html>
345
346 3) http://localhost:8080/demo/bbs/get
347     학생 이름 : 한지민
348     학생 나이 : 25
349
350
351
352 -----
353 Task4. 다양한 GET Request 처리하기
354 1. Package Explorer > right-click > New > Spring Legacy Project
355 2. Select Spring MVC Project
356 3. Project name : MVCDemo
357 4. Next
358 5. Enter a topLevelPackage : com.example.biz
359 6. Finish
360
361
362 7. MVCDemo project 수정하기
363 1) HelloWorldWeb > right-click > Properties
364 2) Java Compiler > JDK Compliance > Compiler compliance level : 13 으로 설정
365 3) Apply click
366 4) Build the project now? > Yes click
367 5) Java Build Path > Libraries tab > Edit click > Select [Workspace default JRE (jdk-13.0.2)]
368    > Finish
369 6) Apply click
370 7) Project Facets > Java > 13
371 8) Runtimes tab > Check [Apache Tomcat v9.0]
372 9) Apply and Close click
373 10) Build the project now? > Yes click
374
375 8. pom.xml 수정하기
376 <properties>
377     <java-version>13</java-version>
378     <org.springframework-version>5.2.5.RELEASE</org.springframework-version>
379     <org.aspectj-version>1.9.5</org.aspectj-version>
380     <org.slf4j-version>1.7.30</org.slf4j-version>
381 </properties>
382 ...
383 <dependency>
384     <groupId>log4j</groupId>
385     <artifactId>log4j</artifactId>
386     <version>1.2.17</version>
387 ...
388 <dependency>
389     <groupId>javax.servlet</groupId>
390     <artifactId>javax.servlet-api</artifactId>
391     <version>4.0.1</version>
392     <scope>provided</scope>
393 </dependency>
394 <dependency>
```

```

395     <groupId>javax.servlet.jsp</groupId>
396     <artifactId>javax.servlet.jsp-api</artifactId>
397     <version>2.3.3</version>
398     <scope>provided</scope>
399 </dependency>
400 <dependency>
401     <groupId>org.junit.jupiter</groupId>
402     <artifactId>junit-jupiter-api</artifactId>
403     <version>5.6.2</version>
404     <scope>test</scope>
405 </dependency>
406
407
408 9. pom.xml > right-click > Run As > Maven install
409 [INFO] BUILD SUCCESS
410
411
412 10. src/main/java/com.example.biz/RequestController.java 생성
413
414     package com.example.biz;
415
416     import org.springframework.stereotype.Controller;
417
418     @Controller
419     public class RequestController {
420     }
421
422
423 11. HttpServletRequest class 이용하기
424     1)src/main/webapp/static folder 생성
425         -src/main/webapp > right-click > New > Folder
426         -Folder name : static
427         -Finish
428
429     2)static folder resource 등록하기
430         -servlet-context.xml에 다음 코드 추가
431         <resources mapping="/static/**" location="/static/" />
432
433     3)src/main/webapp/static/register.html
434
435     <!DOCTYPE html>
436     <html lang="en">
437     <head>
438         <meta charset="UTF-8">
439         <meta name="viewport" content="width=device-width, initial-scale=1.0">
440         <title>회원정보</title>
441     </head>
442     <body>
443         <h1>회원정보</h1>
444         <form action="/biz/confirm" method="GET">
445         <ul>
446             <li>User ID : <input type="text" name="userid"></li>
447             <li>Password : <input type="password" name="passwd"></li>
448             <li>Name : <input type="text" name="name"></li>
449             <li>Age : <input type="number" name="age"></li>
450             <li>Gender : <input type="radio" name="gender" value="남성">남성
451             &nbsp;&nbsp;&nbsp;<input type="radio" name="gender" value="여성">여성</li>
452             <li><input type="submit" value="전송하기"></li>

```



```

452     </ul>
453     </form>
454 </body>
455 </html>
456
457 4)RequestController.java
458
459 @RequestMapping(value="/confirm", method=RequestMethod.GET)
460 public String confirm(HttpServletRequest request, Model model) {
461     String userid = request.getParameter("userid");
462     String passwd = request.getParameter("passwd");
463     String name = request.getParameter("name");
464     int age = Integer.parseInt(request.getParameter("age"));
465     String gender = request.getParameter("gender");
466
467     model.addAttribute("userid", userid);
468     model.addAttribute("passwd", passwd);
469     model.addAttribute("name", name);
470     model.addAttribute("age", age);
471     model.addAttribute("gender", gender);
472     return "confirm"; // /WEB-INF/views/confirm.jsp
473 }
474
475 5)src/main/webapp/WEB-INF/views/confirm.jsp
476
477 <%@ page language="java" contentType="text/html; charset=UTF-8"
478 pageEncoding="UTF-8"%>
479 <!DOCTYPE html>
480 <html>
481     <head>
482         <meta charset="UTF-8">
483         <title>Insert title here</title>
484     </head>
485     <body>
486         아이디 : ${userid} <br />
487         패스워드 : ${passwd} <br />
488         사용자 이름 : ${name} <br />
489         나이 : ${age} <br />
490         성별 : ${gender} <br />
491     </body>
492 </html>
493
494 6)Project right-click > Run As > Run on Server > restart
495 7)<a href="http://localhost:8080/biz/static/register.html">http://localhost:8080/biz/static/register.html
496 8)or <a href="http://localhost:8080/biz/confirm?name=한지민&gender=여성
497    &age=25&userid=jimin&passwd=1234">http://localhost:8080/biz/confirm?name=한지민&gender=여성
498    &age=25&userid=jimin&passwd=1234
499    아이디 : jimin
500    패스워드 : 1234
501    사용자 이름 : 한지민
502    나이 : 25
503    성별 : 여성
504
505 12. @RequestParam Annotation 이용하기
506 1)RequestController.java
507 @RequestMapping(value="/confirm", method=RequestMethod.GET)
508 public String confirm(@RequestParam("userid") String userid,
509 @RequestParam("passwd") String passwd,

```

```

508         @RequestParam("name") String name,
509         @RequestParam("age") int age,
510         @RequestParam("gender") String gender ,Model model) {
511
512         model.addAttribute("userid", userid);
513         model.addAttribute("passwd", passwd);
514         model.addAttribute("name", name);
515         model.addAttribute("age", age);
516         model.addAttribute("gender", gender);
517         return "confirm";
518     }

```

2)src/main/webapp/WEB-INF/views/confirm.jsp

```

522     <%@ page language="java" contentType="text/html; charset=UTF-8"
523     pageEncoding="UTF-8"%>
524     <!DOCTYPE html>
525     <html>
526     <head>
527     <meta charset="UTF-8">
528     <title>Insert title here</title>
529     </head>
530     <body>
531     아이디 : ${userid} <br />
532     패스워드 : ${passwd} <br />
533     사용자 이름 : ${name} <br />
534     나이 : ${age} <br />
535     성별 : ${gender} <br />
536     </body>
537     </html>

```

3)localhost:8080/biz/confirm?name=한지민&gender=여성
&age=25&userid=jimin&passwd=1234

```

539     아이디 : jimin
540     패스워드 : 1234
541     사용자 이름 : 한지민
542     나이 : 25
543     성별 : 여성

```

13. Data Commander 객체 이용하기1

1)lombok library를 pom.xml에 추가하고 maven install할 것

2)src/main/java/com.example.vo.UserVO.java 생성

```

550     -src/main/java > right-click > New > Package
551     -Name : com.example.vo
552     -Finish
553     -com.example.vo > right-click > New > Click
554     -Name : UserVO
555
556     package com.example.vo;
557
558     import lombok.AllArgsConstructor;
559     import lombok.Getter;
560     import lombok.NoArgsConstructor;
561     import lombok.Setter;
562     import lombok.ToString;
563

```

```

564     @NoArgsConstructor
565     @AllArgsConstructor
566     @Getter
567     @Setter
568     @ToString
569     public class UserVO {
570         private String userid;
571         private String passwd;
572         private String name;
573         private int age;
574         private String gender;
575     }
576
577 3)RequestController.java
578
579     @RequestMapping(value="/confirm", method=RequestMethod.GET)
580     public String confirm(@RequestParam("userid") String userid,
581         @RequestParam("passwd") String passwd,
582         @RequestParam("name") String name,
583         @RequestParam("age") int age,
584         @RequestParam("gender") String gender ,Model model) {
585
586         UserVO userVO = new UserVO();
587         userVO.setUserid(userid);
588         userVO.setPasswd(passwd);
589         userVO.setName(name);
590         userVO.setAge(age);
591         userVO.setGender(gender);
592
593         model.addAttribute("userVO", userVO);
594
595         return "confirm1";
596     }
597
598 4)src/main/webapp/WEB-INF/views/confirm1.jsp
599
600     <%@ page language="java" contentType="text/html; charset=UTF-8"
601     pageEncoding="UTF-8"%>
602     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
603     <c:set var="user" value="${userVO}"/>
604     <!DOCTYPE html>
605     <html>
606     <head>
607     <meta charset="UTF-8">
608     <title>Insert title here</title>
609     </head>
610     <body>
611     <h1>confirm1.jsp</h1>
612     <h2>사용자 정보</h2>
613     아이디 : ${user.userid} <br />
614     패스워드 : ${user.passwd} <br />
615     이름 : ${user.name} <br />
616     나이 : ${user.age} <br />
617     성별 : ${user.gender}
618     </body>
619     </html>
620
620 5)Project right-click > Run As > Run on Server > restart

```

```
621 6)localhost:8080/biz/confirm?name=한지민&gender=여성
622    &age=25&userid=jimin&passwd=1234
623    confirm1.jsp
624    사용자 정보
625
626    아이디 : jimin
627    비밀번호 : 1234
628    사용자 이름 : 한지민
629    나이 : 25
630    성별 : 여성
631
632
633 14. Data Commander 객체 이용하기2
634 1)RequestController.java
635
636    @RequestMapping(value="/confirm", method=RequestMethod.GET)
637    public String confirm(UserVO userVO) {
638        return "confirm2";
639    }
640
641 2)src/main/webapp/WEB-INF/views/confirm2.jsp
642
643    <%@ page language="java" contentType="text/html; charset=UTF-8"
644    pageEncoding="UTF-8"%>
645    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
646    <c:set var="user" value="${userVO}"/>
647    <!DOCTYPE html>
648    <html>
649    <head>
650    <meta charset="UTF-8">
651    <title>Insert title here</title>
652    </head>
653    <body>
654    <h1>confirm2.jsp</h1>
655    <h2>사용자 정보</h2>
656    아이디 : ${user.userid} <br />
657    비밀번호 : ${user.passwd} <br />
658    이름 : ${user.name} <br />
659    나이 : ${user.age} <br />
660    성별 : ${user.gender}
661    </body>
662    </html>
663
664 3)Project right-click > Run As > Run on Server > restart
665 4)localhost:8080/biz/confirm?name=한지민&gender=여성
666    &age=25&userid=jimin&passwd=1234
667    confirm2.jsp
668
669    사용자 정보
670
671    아이디 : jimin
672    비밀번호 : 1234
673    사용자 이름 : 한지민
674    나이 : 25
675    성별 : 여성
```

```

676 15. @PathVariable 이용하기
677 1)RequestController.java
678
679     @RequestMapping(value="/confirm/{userid}/{passwd}/{name}/{age}/{gender}",
        method=RequestMethod.GET)
680     public String confirm(@PathVariable String userid, @PathVariable String passwd,
681                          @PathVariable String name, @PathVariable int age,
682                          @PathVariable String gender, Model model) {
683         model.addAttribute("userInfo", new UserVO(userid, passwd, name, age, gender));
684         return "confirm3";
685     }
686
687 2)src/main/webapp/WEB-INF/views/confirm3.jsp
688
689     <%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
690     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
691     <c:set var="user" value="${userInfo}"/>
692     <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
        "http://www.w3.org/TR/html4/loose.dtd">
693     <html>
694     <head>
695     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
696     <title>Insert title here</title>
697     </head>
698     <body>
699         <h1>confirm3.jsp</h1>
700         <h2>사용자 정보</h2>
701         아이디 : ${user.userid} <br />
702         패스워드 : ${user.passwd} <br />
703         이름 : ${user.name} <br />
704         나이 : ${user.age} <br />
705         성별 : ${user.gender}
706     </body>
707     </html>
708
709 3)Project right-click > Run As > Run on Server > restart
710 4)localhost:8080/biz/confirm/jimin/1234/한지민/25/여성
711     confirm3.jsp
712
713     사용자 정보
714
715     아이디 : jimin
716     패스워드 : 1234
717     사용자 이름 : 한지민
718     나이 : 25
719     성별 : 여성
720
721
722
723 -----
724 Task5. @RequestMapping Parameter 다루기
725 1. GET 방식과 POST 방식
726 1)src/main/java/com.example.biz/HomeController.java
727
728     @RequestMapping(value="/login", method=RequestMethod.POST)
729     public String login(@RequestParam("userid") String userid,
730                       @RequestParam("passwd") String passwd,

```

```

731         Model model) {
732
733         model.addAttribute("userid", userid);
734         model.addAttribute("passwd", passwd);
735         return "login";
736     }
737
738 2)src/main/webapp/resources/login.html
739 <!DOCTYPE html>
740 <html>
741 <head>
742 <meta charset="UTF-8">
743 <title>로그인 폼</title>
744 </head>
745 <body>
746     <form method="GET" action="/biz/login">
747         아이디 : <input type="text" name="userid" /><br />
748         패스워드 : <input type="password" name="passwd" /><br />
749         <input type="submit" value="로그인하기" />
750     </form>
751 </body>
752 </html>
753
754 3)<a href="http://localhost:8080/biz/resources/login.html">http://localhost:8080/biz/resources/login.html</a>에서 submit 하면 405 error 발생
755 4)왜냐하면 서로의 method가 불일치하기 때문
756 5)해결방법
757 -src/main/java/com.example.biz/HomeController.java 수정
758 -즉 login method(요청 처리 method)의 이름은 같지만 parameter의 type과 return type이 틀리기 때문
   에 Method Overloading 됨.
759
760     @RequestMapping(value="/login", method=RequestMethod.POST)
761     public String login(@RequestParam("userid") String userid,
762                        @RequestParam("passwd") String passwd,
763                        Model model) {
764
765         model.addAttribute("userid", userid);
766         model.addAttribute("passwd", passwd);
767         return "login";
768     }
769     @RequestMapping(value="/login", method=RequestMethod.GET)
770     public ModelAndView login(@RequestParam("userid") String userid,
771                             @RequestParam("passwd") String passwd) {
772
773         ModelAndView mav = new ModelAndView();
774         mav.addObject("userid", userid);
775         mav.addObject("passwd", passwd);
776         mav.setViewName("login");
777         return mav;
778     }
779
780 6)src/main/webapp/WEB-INF/views/login.jsp
781
782 <%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
783 <!DOCTYPE html>
784 <html>
785 <head>
786 <meta charset="UTF-8">

```

```
787 <title>Insert title here</title>  
788 </head>  
789 <body>  
790     아이디 : ${userid} <br />  
791     패스워드 : ${passwd}  
792 </body>  
793 </html>  
794  
795 7) http://localhost:8080/biz/resources/login.html  
796     아이디 : jimin  
797     패스워드 : 1234  
798  
799  
800 2. @ModelAttribute Annotation 이용하기  
801 1)@ModelAttribute Annotation을 이용하면 Data Commander 객체의 이름을 변경할 수 있다.  
802 2)src/main/webapp/resources/register.html  
803  
804 <!DOCTYPE html>  
805 <html>  
806 <head>  
807 <meta charset="UTF-8">  
808 <title>회원가입 폼</title>  
809 </head>  
810 <body>  
811     <form method="POST" action="/biz/register">  
812         아이디 : <input type="text" name="userid" /><br />  
813         패스워드 : <input type="password" name="passwd" /><br />  
814         이름 : <input type="text" name="name" /><br />  
815         나이 : <input type="number" name="age" /><br />  
816         성별 : <input type="radio" name="gender" value="남성" />남성 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~  
817             <input type="radio" name="gender" value="여성" />여성<br />  
818             <input type="submit" value="가입하기" />  
819     </form>  
820 </body>  
821 </html>  
822  
823 3)src/main/java/com.example.biz/HomeController.java  
824  
825 @RequestMapping(value="/register", method=RequestMethod.POST)  
826 public String register(@ModelAttribute("u") UserVO userVO) {    //userVO가 아니라 u로 변경  
827  
828     return "register";  
829 }  
830  
831 4)src/main/webapp/WEB-INF/views/register.jsp  
832  
833 <%@ page language="java" contentType="text/html; charset=UTF-8"  
pageEncoding="UTF-8"%>  
834 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
835 <c:set var="user" value="${u}" />  
836 <!DOCTYPE html>  
837 <html>  
838 <head>  
839 <meta charset="UTF-8">  
840 <title>Insert title here</title>  
841 </head>  
842 <body>  
843     <h1>사용자 정보</h1>
```

```

844     <ul>
845         <li>아이디 : ${user.userid}</li>
846         <li>패스워드 : ${user.passwd}</li>
847         <li>이름 : ${user.name}</li>
848         <li>나이 : ${user.age}</li>
849         <li>성별 : ${user.gender}</li>
850     </ul>
851 </body>
852 </html>
853
854 5)Spring에서 POST 방식으로 Data를 보낼 때 한글깨짐 현상 발생
855 6)해결방법
856 7)web.xml
857
858     <filter>
859         <filter-name>encodingFilter</filter-name>
860         <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
861         <init-param>
862             <param-name>encoding</param-name>
863             <param-value>UTF-8</param-value>
864         </init-param>
865     </filter>
866     <filter-mapping>
867         <filter-name>encodingFilter</filter-name>
868         <url-pattern>/*</url-pattern>
869     </filter-mapping>
870
871 8)<a href="http://localhost:8080/biz/resources/register.html">http://localhost:8080/biz/resources/register.html</a> -->
872 9)<a href="http://localhost:8080/biz/register">http://localhost:8080/biz/register</a>
873     사용자 정보
874
875     아이디 : jimin
876     패스워드 : 1234
877     사용자 이름 : 한지민
878     나이 : 25
879     성별 : 여성
880
881
882 3. redirect: 키워드 이용하기
883 1)src/main/java/com.example.biz/HomeController.java
884
885     @RequestMapping("/verify")
886     public String verify(HttpServletRequest request, Model model) {
887         String userid = request.getParameter("userid");
888         if(userid.equals("admin")) {    //만일 userid가 admin 이면 /admin으로 리다이렉트
889             return "redirect:admin";
890         }
891         return "redirect:user";        //만일 userid가 admin 이 아니면 /user로 리다이렉트
892         //return "redirect:<a href="http://www.naver.com">http://www.naver.com</a>";    //절대 경로도 가능
893     }
894
895     @RequestMapping("/admin")
896     public String verify1(Model model) {
897         model.addAttribute("authority", "관리자권한");
898         return "admin";
899     }
900
901     @RequestMapping("/user")

```



```
902     public String verify2(Model model) {
903         model.addAttribute("authority", "일반사용자");
904         return "user";
905     }
906
907 2)/src/main/webapp/WEB-INF/views/admin.jsp
908 <%@ page language="java" contentType="text/html; charset=UTF-8"
909     pageEncoding="UTF-8"%>
910 <!DOCTYPE html>
911 <html>
912 <head>
913 <meta charset=UTF-8">
914 <title>Insert title here</title>
915 </head>
916 <body>
917     <h1>관리자 페이지</h1>
918     권한 : ${authority}
919 </body>
920 </html>
921
922 3)/src/main/webapp/WEB-INF/views/user.jsp
923
924 <%@ page language="java" contentType="text/html; charset=UTF-8"
925     pageEncoding="UTF-8"%>
926 <!DOCTYPE html>
927 <html>
928 <head>
929 <meta charset=UTF-8">
930 <title>Insert title here</title>
931 </head>
932 <body>
933     <h1>일반 사용자 페이지</h1>
934     권한 : ${authority}
935 </body>
936 </html>
937
938 4)<a href="http://localhost:8080/biz/verify?userid=admin">http://localhost:8080/biz/verify?userid=admin --> <a href="http://localhost:8080/biz/admin">http://localhost:8080/biz/admin
939 5)<a href="http://localhost:8080/biz/verify?userid=user">http://localhost:8080/biz/verify?userid=user --> <a href="https://www.naver.com">https://www.naver.com
940
941 -----
942 Task6. Java Annotation으로 Web Application project 생성하기
943 1. Package Explorer > right-click > New > Spring Legacy Project
944 2. Select Spring MVC Project
945 3. Project name : HelloWorldWeb1
946 4. Next
947 5. Enter a topLevelPackage : com.example.biz
948 6. Finish
949
950
951 7. HelloWorldWeb1 project 수정하기
952 1)HelloWorldWeb > right-click > Properties
953 2)Java Compiler > JDK Compliance > Compiler compliance level : 13 으로 설정
954 3)Apply click
955 4)Build the project now? > Yes click
956 5)Java Build Path > Libraries tab > Edit click > Select [Workspace default JRE (jdk-13.0.2)]
957 > Finish
```

```
957 6)Apply click
958 7)Project Facets > Java > 13
959 8)Runtimes tab > Check [Apache Tomcat v9.0]
960 9)Apply and Close click
961 10)Build the project now? > Yes click
962
963
964 8. pom.xml 수정하기
965 <properties>
966     <java-version>13</java-version>
967     <org.springframework-version>5.2.5.RELEASE</org.springframework-version>
968     <org.aspectj-version>1.9.5</org.aspectj-version>
969     <org.slf4j-version>1.7.30</org.slf4j-version>
970 </properties>
971 ...
972 <dependency>
973     <groupId>log4j</groupId>
974     <artifactId>log4j</artifactId>
975     <version>1.2.17</version>
976 ...
977 <dependency>
978     <groupId>javax.servlet</groupId>
979     <artifactId>javax.servlet-api</artifactId>
980     <version>4.0.1</version>
981     <scope>provided</scope>
982 </dependency>
983 <dependency>
984     <groupId>javax.servlet.jsp</groupId>
985     <artifactId>javax.servlet.jsp-api</artifactId>
986     <version>2.3.3</version>
987     <scope>provided</scope>
988 </dependency>
989 <dependency>
990     <groupId>org.junit.jupiter</groupId>
991     <artifactId>junit-jupiter-api</artifactId>
992     <version>5.6.2</version>
993     <scope>test</scope>
994 </dependency>
995
996
997 9. pom.xml > right-click > Run As > Maven install
998 [INFO] BUILD SUCCESS
999
1000
1001 10. HelloWorldWeb1 > right-click > Run As > Run on Server
1002 Hello world!
1003 The time on the server is April 19, 2020 at 10:16:40 PM KST.
1004
1005
1006 11. web.xml 삭제
1007 12. /WEB-INF/spring folder 삭제
1008 13. pom.xml 수정
1009 1)web.xml을 삭제하면 pom.xml에서 error 발생
1010 2)pom.xml 하단에 <plugins> 설정
1011
1012 <plugin>
1013     <groupId>org.apache.maven.plugins</groupId>
1014     <artifactId>maven-war-plugin</artifactId>
```

```

1015     <version>3.2.3</version>
1016     <configuration>
1017         <failOnMissingWebXml>false</failOnMissingWebXml>
1018     </configuration>
1019 </plugin>
1020
1021
1022 14. com.example.config folder 생성
1023 1)src/main/java > right-click > New > Folder
1024 2)Folder name : config
1025 3)Finish
1026 4)config > right-click > New > Class
1027 5)Name : RootConfig
1028 6)Finish
1029
1030     package config;
1031
1032     import org.springframework.context.annotation.Configuration;
1033
1034     @Configuration
1035     public class RootConfig {
1036
1037     }
1038
1039 7)config > right-click > New > Class
1040 8)Name : ServletConfig
1041 9)Interfaces : org.springframework.web.servlet.config.annotation.WebMvcConfigurer
1042 10)Finish
1043
1044     package config;
1045
1046     import org.springframework.context.annotation.ComponentScan;
1047     import org.springframework.web.servlet.config.annotation.EnableWebMvc;
1048     import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
1049     import org.springframework.web.servlet.config.annotation.ViewResolverRegistry;
1050     import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
1051     import org.springframework.web.servlet.view.InternalResourceViewResolver;
1052     import org.springframework.web.servlet.view.JstlView;
1053
1054     @EnableWebMvc
1055     @ComponentScan(basePackages = {"com.example"})
1056     public class ServletConfig implements WebMvcConfigurer {
1057
1058         @Override
1059         public void configureViewResolvers(ViewResolverRegistry registry) {
1060             InternalResourceViewResolver bean = new InternalResourceViewResolver();
1061             bean.setViewClass(JstlView.class);
1062             bean.setPrefix("/WEB-INF/views/");
1063             bean.setSuffix(".jsp");
1064             registry.viewResolver(bean);
1065         }
1066         @Override
1067         public void addResourceHandlers(ResourceHandlerRegistry registry) {
1068             registry.addResourceHandler("/resources/**").addResourceLocations("/resources/");
1069         }
1070     }
1071
1072 11)config > right-click > New > Class

```

```
1073 12)Name : WebConfig
1074
1075     package config;
1076
1077     import
1078     org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitia
1079     lizer;
1080
1081     public class WebConfig extends AbstractAnnotationConfigDispatcherServletInitializer{
1082
1083         @Override
1084         protected Class<?>[] getRootConfigClasses() {
1085             return new Class[] { RootConfig.class };
1086         }
1087
1088         @Override
1089         protected Class<?>[] getServletConfigClasses() {
1090             return new Class[] { ServletConfig.class };
1091         }
1092
1093         @Override
1094         protected String[] getServletMappings() {
1095             return new String[] { "/" };
1096         }
1097     }
```

1098 15. HelloWorldWeb1 > right-click > Run As > Run on Server > restart

1099 Hello world!

1100
1101 The time on the server is April 19, 2020 at 10:39:26 PM KST.

1102

1103

1104 16. ServletConfig.java 수정

```
1105     @Override
1106     public void addResourceHandlers(ResourceHandlerRegistry registry) {
1107         registry.addResourceHandler("/resources/**").addResourceLocations("/resources/");
1108         registry.addResourceHandler("/images/**").addResourceLocations("/images/");
1109         registry.addResourceHandler("/static/**").addResourceLocations("/static/");
1110     }
```

1111

1112

1113 17. webapp/static folder 생성

1114 1)webapp/static/css folder 생성

1115 -bootstrap-theme.css

1116 -bootstrap.css

1117

1118 2)webapp/static/fonts folder 생성

1119 -glyphicons-halflings-regular.eot

1120 -glyphicons-halflings-regular.svg

1121 -glyphicons-halflings-regular.ttf

1122 -glyphicons-halflings-regular.woff

1123 -glyphicons-halflings-regular.woff2

1124

1125 3)webapp/static/js folder 생성

1126 -bootstrap.js

1127 -jquery-3.4.1.js

1128

```
1129
1130 18. webapp/images folder 생성
1131     1)apple.jpg
1132
1133
1134 19. views/main.jsp
1135
1136 <%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
1137 <!DOCTYPE html>
1138 <html lang="en">
1139 <head>
1140     <meta charset="UTF-8">
1141     <meta name="viewport" content="width=device-width, initial-scale=1.0">
1142     <title>Welcome example.com</title>
1143     <link rel="stylesheet" href="static/css/bootstrap.css">
1144     <link rel="stylesheet" href="static/css/bootstrap-theme.css">
1145     <script src="static/js/jquery-3.4.1.js"></script>
1146 </head>
1147 <body>
1148     <div class="jumbotron">
1149         <h1>Welcome to www.example.com</h1>
1150         <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nemo accusantium,
aspernatur quos porro commodi perspiciatis, assumenda consequuntur eius inventore
eaque omnis, magni natus corrupti doloremque? Quia aliquid excepturi tempora
praesentium modi necessitatibus sequi quisquam dolorem nihil deserunt magnam,
distinctio sunt aspernatur nisi. Id odio quaerat amet quidem adipisci totam ad.</p>
1151     <p>
1152         <h2>Example heading <span class="label label-success">Success</span></h2>
1153         <h4>Current Server Time is <span class="label
label-info">${serverTime}</span></h4>
1154     </p>
1155     </div>
1156     <div>
1157         
1158     </div>
1159 </body>
1160 </html>
1161
1162
1163 20. com.example.biz/HomeController.java 수정
1164
1165 package com.example.biz;
1166
1167 import java.text.DateFormat;
1168 import java.util.Date;
1169 import java.util.Locale;
1170
1171 import org.slf4j.Logger;
1172 import org.slf4j.LoggerFactory;
1173 import org.springframework.stereotype.Controller;
1174 import org.springframework.ui.Model;
1175 import org.springframework.web.bind.annotation.RequestMapping;
1176 import org.springframework.web.bind.annotation.RequestMethod;
1177
1178 /**
1179  * Handles requests for the application home page.
1180  */
```

```

1181 @Controller
1182 public class HomeController {
1183     private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
1184     @RequestMapping(value = "/", method = RequestMethod.GET)
1185     public String home(Locale locale, Model model) {
1186         logger.info("Welcome home! The client locale is {}. ", locale);
1187
1188         Date date = new Date();
1189         DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG,
1190             DateFormat.LONG, locale);
1191         String formattedDate = dateFormat.format(date);
1192         model.addAttribute("serverTime", formattedDate );
1193         return "main";
1194     }
1195 }
1196
1197
1198 21. HelloWorldWeb1 > right-click > Run As > Run on Server > restart
1199
1200
1201
1202 -----
1203 Task7. Database와 연동하기
1204 1. Package Explorer > right-click > New > Spring Legacy Project
1205 2. Select Spring MVC Project
1206 3. Project name : MVCDemo1
1207 4. Next
1208 5. Enter a topLevelPackage : com.example.biz
1209 6. Finish
1210
1211
1212 7. MVCDemo1 project 수정하기
1213 1)MVCDemo1 > right-click > Properties
1214 2)Java Compiler > JDK Compliance > Compiler compliance level : 13 으로 설정
1215 3)Apply click
1216 4)Build the project now? > Yes click
1217 5)Java Build Path > Libraries tab > Edit click > Select [Workspace default JRE (jdk-13.0.2)]
1218 > Finish
1219 6)Apply click
1220 7)Project Facets > Java > 13
1221 8)Runtimes tab > Check [Apache Tomcat v9.0]
1222 9)Apply and Close click
1223 10)Build the project now? > Yes click
1224
1225 8. pom.xml 수정하기
1226 <properties>
1227     <java-version>13</java-version>
1228     <org.springframework-version>5.2.5.RELEASE</org.springframework-version>
1229     <org.aspectj-version>1.9.5</org.aspectj-version>
1230     <org.slf4j-version>1.7.30</org.slf4j-version>
1231 </properties>
1232 ...
1233 <dependency>
1234     <groupId>log4j</groupId>
1235     <artifactId>log4j</artifactId>
1236     <version>1.2.17</version>

```

```

1237 ...
1238 <dependency>
1239     <groupId>javax.servlet</groupId>
1240     <artifactId>javax.servlet-api</artifactId>
1241     <version>4.0.1</version>
1242     <scope>provided</scope>
1243 </dependency>
1244 <dependency>
1245     <groupId>javax.servlet.jsp</groupId>
1246     <artifactId>javax.servlet.jsp-api</artifactId>
1247     <version>2.3.3</version>
1248     <scope>provided</scope>
1249 </dependency>
1250 <dependency>
1251     <groupId>org.junit.jupiter</groupId>
1252     <artifactId>junit-jupiter-api</artifactId>
1253     <version>5.6.2</version>
1254     <scope>test</scope>
1255 </dependency>
1256
1257
1258 9. pom.xml > right-click > Run As > Maven install
1259 [INFO] BUILD SUCCESS
1260
1261
1262 10. Create Table in MariaDB
1263 CREATE TABLE Member
1264 (
1265     userid      VARCHAR(20),
1266     username    VARCHAR(20) NOT NULL,
1267     userage     TINYINT  NOT NULL,
1268     gender      VARCHAR(10) NOT NULL,
1269     city        VARCHAR(50),
1270     CONSTRAINT member_userid_pk PRIMARY KEY(userid)
1271 );
1272 -반드시 [test] Database의 조합을 utf8_general_ci로 맞출 것
1273 -반드시 Member Table의 기본조합이 utf8_general_ci 임을 확인할 것
1274
1275
1276 11. src/main/webapp/static folder 생성
1277 1)src/main/webapp/static/css folder
1278 2)src/main/webapp/static/images folder
1279 3)src/main/webapp/static/js folder
1280 -jquery-1.12.4.js
1281 4)src/main/webapp/static/register.html
1282 <!DOCTYPE html>
1283 <html lang="en">
1284 <head>
1285     <meta charset="UTF-8">
1286     <title>회원 가입</title>
1287 </head>
1288 <body>
1289     <h1>회원 가입 창</h1>
1290     <form action="/biz/create" method="post">
1291         <ul>
1292             <li>ID : <input type="text" name="userid" /></li>
1293             <li>이름 : <input type="text" name="username" /></li>
1294             <li>나이 : <input type="number" name="age" /></li>

```

```

1295         <li>성별 : <input type="radio" name="gender" value="남성"/>남성
1296             <input type="radio" name="gender" value="여성"/>여성</li>
1297         <li>거주지 : <input type="text" name="city" /></li>
1298         <li><input type="submit" value="가입하기" /></li>
1299     </ul>
1300 </form>
1301 </body>
1302 </html>
1303
1304
1305 10. src/main/webapp/WEB-INF/spring/appServlet/sevlet-context.xml 수정
1306     <resources mapping="/static/**" location="/static/" /> 추가
1307
1308     <context:component-scan base-package="com.example" /> 수정
1309
1310
1311 11. src/main/resources/mariadb.properties
1312     db.driverClass=org.mariadb.jdbc.Driver
1313     db.url=jdbc:mariadb://localhost:3306/test
1314     db.username=root
1315     db.password=javamariadb
1316
1317
1318 12. Spring JDBC 설치
1319     1)JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
1320
1321     <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
1322     <dependency>
1323         <groupId>org.springframework</groupId>
1324         <artifactId>spring-jdbc</artifactId>
1325         <version>5.2.0.RELEASE</version>
1326     </dependency>
1327
1328     2)pom.xml에 붙여 넣고 Maven Install 하기
1329     [INFO] BUILD SUCCESS
1330
1331
1332 13. MariaDB Jdbc Driver library 검색 및 설치
1333     1)Maven Repository 에서 'mariadb'로 검색하여 MariaDB Java Client를 설치한다.
1334
1335     <dependency>
1336         <groupId>org.mariadb.jdbc</groupId>
1337         <artifactId>mariadb-java-client</artifactId>
1338         <version>2.5.1</version>
1339     </dependency>
1340
1341     2)pom.xml에 붙여 넣고 Maven Install 하기
1342     [INFO] BUILD SUCCESS
1343
1344
1345 14. src/main/webapp/WEB-INF/spring/root-context.xml
1346     <?xml version="1.0" encoding="UTF-8"?>
1347     <beans xmlns="http://www.springframework.org/schema/beans"
1348         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1349         xmlns:context="http://www.springframework.org/schema/context"
1350         xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context

```


<http://www.springframework.org/schema/context/spring-context-4.3.xsd>">

```

1352
1353 <!-- Root Context: defines shared resources visible to all other web components -->
1354 <context:property-placeholder location="classpath:mariadb.properties"/>
1355 <bean id="dataSource"
1356     class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
1357     <property name="driverClass" value="${db.driverClass}" />
1358     <property name="url" value="${db.url}" />
1359     <property name="username" value="${db.username}" />
1360     <property name="password" value="${db.password}" />
1361 </bean>
1362 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
1363     <property name="dataSource" ref="dataSource" />
1364 </bean>
1365 </beans>
1366
1367
1368 15. src/test/java/com.example.biz/TestApp class 생성
1369 1)com.example.biz > right-click > New > JUnit Test Case
1370 2)Select [New JUnit 4 test]
1371 3)Name : TestApp
1372 4)Finish
1373     package com.example.biz;
1374
1375     import org.junit.Before;
1376     import org.junit.Test;
1377     import org.springframework.context.ApplicationContext;
1378     import org.springframework.context.support.GenericXmlApplicationContext;
1379     import org.springframework.jdbc.core.JdbcTemplate;
1380
1381     public class TestApp {
1382         private ApplicationContext ctx;
1383
1384         @Before
1385         public void init() {
1386             this.ctx = new
1387                 GenericXmlApplicationContext("file:src/main/webapp/WEB-INF/spring**/root-context
1388                 .xml");
1389         }
1390         @Test
1391         public void test() {
1392             JdbcTemplate jdbcTemplate = this.ctx.getBean("jdbcTemplate", JdbcTemplate.class);
1393             System.out.println(jdbcTemplate);
1394         }
1395     }
1396
1397 5)Run as > JUnit Test > Green bar
1398
1399 16. package 생성
1400 1)src/main/java/com.example.vo
1401 2)src/main/java/com.example.dao
1402 3)src/main/java/com.example.service
1403
1404 17. src/com.example.vo.MemberVO class 생성
1405

```

```
1406 package com.example.vo;
1407
1408 public class MemberVO {
1409     private String userid;
1410     private String username;
1411     private int age;
1412     private String gender;
1413     private String city;
1414
1415     public MemberVO() {}
1416
1417     public MemberVO(String userid, String username, int age, String gender, String city) {
1418         this.userid = userid;
1419         this.username = username;
1420         this.age = age;
1421         this.gender = gender;
1422         this.city = city;
1423     }
1424
1425     public String getUserid() {
1426         return userid;
1427     }
1428
1429     public void setUserid(String userid) {
1430         this.userid = userid;
1431     }
1432
1433     public String getUsername() {
1434         return username;
1435     }
1436
1437     public void setUsername(String username) {
1438         this.username = username;
1439     }
1440
1441     public int getAge() {
1442         return age;
1443     }
1444
1445     public void setAge(int age) {
1446         this.age = age;
1447     }
1448
1449     public String getGender() {
1450         return gender;
1451     }
1452
1453     public void setGender(String gender) {
1454         this.gender = gender;
1455     }
1456
1457     public String getCity() {
1458         return city;
1459     }
1460
1461     public void setCity(String city) {
1462         this.city = city;
1463     }
1464 }
```

```
1464
1465     @Override
1466     public String toString() {
1467         return "MemberVO [userid=" + userid + ", username=" + username + ", age=" + age
1468             + ", gender=" + gender
1469             + ", city=" + city + "]";
1470     }
1471 }
1472 18. com/example.dao
1473 1)MemberDao interface
1474     package com.example.dao;
1475
1476     import java.util.List;
1477
1478     import com.example.vo.MemberVO;
1479
1480     public interface MemberDao {
1481         int create(MemberVO memberVo);
1482         MemberVO read(String userid);
1483         List<MemberVO> readAll();
1484         int update(MemberVO memberVo);
1485         int delete(String userid);
1486     }
1487
1488 2)MemberDaoImpl.java
1489     package com.example.dao;
1490
1491     import java.util.List;
1492
1493     import org.springframework.beans.factory.annotation.Autowired;
1494     import org.springframework.jdbc.core.JdbcTemplate;
1495     import org.springframework.stereotype.Repository;
1496
1497     import com.example.vo.MemberVO;
1498
1499     @Repository("memberDao")
1500     public class MemberDaoImpl implements MemberDao {
1501         @Autowired
1502         JdbcTemplate jdbcTemplate;
1503
1504         @Override
1505         public int create(MemberVO memberVo) {
1506             return 0;
1507         }
1508
1509         @Override
1510         public MemberVO read(String userid) {
1511             return null;
1512         }
1513
1514         @Override
1515         public List<MemberVO> readAll() {
1516             return null;
1517         }
1518
1519         @Override
1520         public int update(MemberVO memberVo) {
```

```
1521     return 0;
1522 }
1523
1524 @Override
1525 public int delete(String userid) {
1526     return 0;
1527 }
1528 }
1529
1530 19. com.example.service
1531 1)MemberService interface
1532     package com.example.service;
1533
1534     import java.util.List;
1535
1536     import com.example.vo.MemberVO;
1537
1538     public interface MemberService {
1539         int create(MemberVO memberVo);
1540         MemberVO read(String userid);
1541         List<MemberVO> readAll();
1542         int update(MemberVO memberVo);
1543         int delete(String userid);
1544     }
1545
1546 2)MemberServiceImpl.java
1547     package com.example.service;
1548
1549     import java.util.List;
1550
1551     import org.springframework.beans.factory.annotation.Autowired;
1552     import org.springframework.stereotype.Service;
1553
1554     import com.example.dao.MemberDao;
1555     import com.example.vo.MemberVO;
1556
1557     @Service("memberService")
1558     public class MemberServiceImpl implements MemberService {
1559         @Autowired
1560         MemberDao memberDao;
1561
1562         @Override
1563         public int create(MemberVO memberVo) {
1564             return 0;
1565         }
1566
1567         @Override
1568         public MemberVO read(String userid) {
1569             return null;
1570         }
1571
1572         @Override
1573         public List<MemberVO> readAll() {
1574             return null;
1575         }
1576
1577         @Override
1578         public int update(MemberVO memberVo) {
```

```
1579         return 0;
1580     }
1581
1582     @Override
1583     public int delete(String userid) {
1584         return 0;
1585     }
1586 }
1587
1588
1589 20. com.example.biz
1590 1)HomeController.java
1591
1592     package com.example.biz;
1593
1594     import org.springframework.beans.factory.annotation.Autowired;
1595     import org.springframework.stereotype.Controller;
1596
1597     import com.example.service.MemberService;
1598
1599     /**
1600      * Handles requests for the application home page.
1601      */
1602     @Controller
1603     public class HomeController {
1604         @Autowired
1605         MemberService memberService;
1606     }
1607
1608
1609 21. Data Insert
1610 1)/src/main/webapp/static/register.html
1611 2)com.example.biz/HomeController.java
1612
1613     @Controller
1614     public class HomeController {
1615         @Autowired
1616         MemberService memberService;
1617
1618         @RequestMapping(value = "/create", method = RequestMethod.POST)
1619         public String home(MemberVO memberVo, Model model) {
1620             int row = this.memberService.create(memberVo);
1621             if(row == 1) model.addAttribute("status", "Insert Success");
1622             else model.addAttribute("status", "Insert Failure");
1623             return "create"; // /WEB-INF/views/create.jsp
1624         }
1625     }
1626
1627 3)com.example.service/MemberServiceImpl.java
1628
1629     @Service("memberService")
1630     public class MemberServiceImpl implements MemberService {
1631         @Autowired
1632         MemberDao memberDao;
1633
1634         @Override
1635         public int create(MemberVO memberVo) {
1636             return this.memberDao.create(memberVo);
```

```
1637     }
1638 }
1639
1640 4)com.example.dao.MemberDaoImpl.java
1641
1642 @Repository("memberDao")
1643 public class MemberDaoImpl implements MemberDao {
1644     @Autowired
1645     JdbcTemplate jdbcTemplate;
1646
1647     @Override
1648     public int create(MemberVO memberVo) {
1649         String sql = "INSERT INTO Member VALUES(?,?,?,?)";
1650         return this.jdbcTemplate.update(sql, memberVo.getUserid(),
1651             memberVo.getUsername(), memberVo.getAge(),
1652             memberVo.getGender(), memberVo.getCity());
1653     }
1654 }
1655
1656 5)views/create.jsp
1657
1658 <%@ page language="java" contentType="text/html; charset=UTF-8"
1659     pageEncoding="UTF-8"%>
1660 <!DOCTYPE html>
1661 <html>
1662 <head>
1663 <meta charset="UTF-8">
1664 <title>Insert title here</title>
1665 </head>
1666 <body>
1667 <h1>${status}</h1>
1668 </body>
1669 </html>
1670
1671 22. POST 발송시 한글 깨짐 처리하기
1672 1)web.xml
1673
1674 <filter>
1675 <filter-name>encodingFilter</filter-name>
1676 <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
1677 <init-param>
1678 <param-name>encoding</param-name>
1679 <param-value>UTF-8</param-value>
1680 </init-param>
1681 </filter>
1682 <filter-mapping>
1683 <filter-name>encodingFilter</filter-name>
1684 <url-pattern>/*</url-pattern>
1685 </filter-mapping>
1686
1687 23. Test
1688 1)<a href="http://localhost:8080/biz/static/register.html">http://localhost:8080/biz/static/register.html
1689 2)<a href="http://localhost:8080/biz/create">http://localhost:8080/biz/create
1690 Insert Success
1691
1692 24. Data Select
1693 1)HomeController.java
```

```
1694
1695     @RequestMapping(value = "/view/{userid}", method = RequestMethod.GET)
1696     public String view(@PathVariable String userid, Model model) {
1697         MemberVO memberVo = this.memberService.read(userid);
1698         model.addAttribute("member", memberVo);
1699         return "view";
1700     }
1701
1702 2)MemberServiceImpl.java
1703
1704     @Override
1705     public MemberVO read(String userid) {
1706         return this.memberDao.read(userid);
1707     }
1708
1709 3)MemberDaoImpl.java
1710
1711     @Override
1712     public MemberVO read(String userid) {
1713         String sql = "SELECT * FROM Member WHERE userid = ?";
1714         return this.jdbcTemplate.queryForObject(sql, new Object[] {userid},
1715             new MyRowMapper());
1716     }
1717
1718     class MyRowMapper implements RowMapper<MemberVO>{
1719         @Override
1720         public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1721             MemberVO memberVo = new MemberVO(rs.getString("userid"),
1722                 rs.getString("username"), rs.getInt("userage"),
1723                 rs.getString("gender"), rs.getString("city"));
1724             return memberVo;
1725         }
1726     }
1727
1728 4)views/view.jsp
1729
1730     <%@ page language="java" contentType="text/html; charset=UTF-8"
1731     pageEncoding="UTF-8"%>
1732     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
1733     <c:set var="user" value="${member}" />
1734     <!DOCTYPE html>
1735     <html>
1736     <head>
1737         <meta charset="UTF-8">
1738         <title>Insert title here</title>
1739         <script src="/biz/static/js/jquery-1.12.4.js"></script>
1740         <script>
1741             $(function(){
1742                 $("#btnList").bind("click", function(){
1743                     location.href = "/biz/list";
1744                 });
1745                 $("#btnDelete").bind("click", function(){
1746                     location.href = "/biz/delete/${user.userid}";
1747                 });
1748             });
1749         </script>
1750     </head>
1751     <body>
```

```

1751 <h1>${user.username}의 정보</h1>
1752 <form action="/biz/update" method="post">
1753     <input type="hidden" name="userid" value = "${user.userid}" />
1754     <ul>
1755         <li>아이디 : ${user.userid }</li>
1756         <li>나이 : <input type='number' name="age" value='${user.age}' /></li>
1757         <li>성별 : <c:if test='${user.gender eq "남성"}'>
1758             <input type="radio" name="gender" value="남성" checked />남성&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
1759             <input type="radio" name="gender" value="여성" />여성
1760             </c:if>
1761             <c:if test='${user.gender eq "여성"}'>
1762                 <input type="radio" name="gender" value="남성" />남성&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
1763                 <input type="radio" name="gender" value="여성" checked />여성
1764             </c:if>
1765         </li>
1766         <li>거주지 : <input type="text" name="city" value="${user.city }" /></li>
1767         <li><input type='submit' value='수정하기' /></li>
1768         <li><input type='button' value='삭제하기' id="btnDelete"/></li>
1769         <li><input type='button' value='목록으로' id="btnList"/></li>
1770     </ul>
1771 </form>
1772 </body>
1773 </html>

```

5) Test

<http://localhost:8080/biz/view/jimin>

25. Data List

1)HomeController.java

```
@RequestMapping(value = "/list", method = RequestMethod.GET)
public String list(Model model) {
    List<MemberVO> list = this.memberService.readAll();
    model.addAttribute("userlist", list);
    return "list";    // /WEB-INF/views/list.jsp
}
```

2)MemberServiceImpl.java

```
@Override
public List<MemberVO> readAll() {
    return this.memberDao.readAll();
}
```

3)MemberDaoImpl.java

```
@Override
public List<MemberVO> readAll() {
    String sql = "SELECT * FROM Member ORDER BY userid DESC";
    return this.jdbcTemplate.query(sql, new MyRowMapper());
}
```

4)iews/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" %>
```



```

1808 <!DOCTYPE html>
1809 <html>
1810 <head>
1811 <meta charset="UTF-8">
1812 <title>Insert title here</title>
1813 </head>
1814 <body>
1815 <h1>Member List</h1>
1816 <table border='1'>
1817 <thead>
1818 <tr>
1819 <th>아이디</th><th>이름</th><th>나이</th><th>성별</th><th>거주지</th>
1820 </tr>
1821 </thead>
1822 <tbody>
1823 <c:forEach items="${userlist}" var="user">
1824 <tr>
1825 <td><a
1826 href="/biz/view/${user.userid}">${user.userid}</a></td><td>${user.userna
1827 me}</td>
1828 <td>${user.age}</td><td>${user.gender }</td>
1829 <td>${user.city }</td>
1830 </tr>
1831 </c:forEach>
1832 </tbody>
1833 </table>
1834 </body>
1835 </html>

```

5)Test

<http://localhost:8080/biz/list>

26. Data Delete

1)HomeController.java

```

1842 @RequestMapping(value = "/delete/{userid}", method = RequestMethod.GET)
1843 public String delete(@PathVariable String userid) {
1844     this.memberService.delete(userid);
1845     return "redirect:/list";
1846 }

```

2)MemberServiceImpl.java

```

1850 @Override
1851 public int delete(String userid) {
1852     return this.memberDao.delete(userid);
1853 }

```

3)MemberDaoImpl.java

```

1857 @Override
1858 public int delete(String userid) {
1859     String sql = "DELETE FROM Member WHERE userid = ?";
1860     return this.jdbcTemplate.update(sql, userid);
1861 }

```

```
1863 4)Test
1864 http://localhost:8080/biz/delete/chulsu
1865
1866
1867 27. Data Update
1868 1)HomeController.java
1869
1870 @RequestMapping(value = "/update", method = RequestMethod.POST)
1871 public String update(@RequestParam("userid") String userid,
1872     @RequestParam("age") int age,
1873     @RequestParam("gender") String gender,
1874     @RequestParam("city") String city) {
1875     this.memberService.update(
1876         new MemberVO(userid, "", age, gender, city));
1877     return "redirect:/list";
1878 }
1879
1880 2)MemberServiceImpl.java
1881
1882 @Override
1883 public int update(MemberVO memberVo) {
1884     return this.memberDao.update(memberVo);
1885 }
1886
1887 3)MemberDaoImpl.java
1888
1889 @Override
1890 public int update(MemberVO memberVo) {
1891     String sql = "UPDATE Member SET userage = ?, gender = ?, city = ? " +
1892         "WHERE userid = ?";
1893     return this.jdbcTemplate.update(sql, memberVo.getAge(),
1894         memberVo.getGender(), memberVo.getCity(), memberVo.getUserid());
1895 }
1896
1897 4)Test
1898 http://localhost:8080/biz/list에서
1899 해당 ID Click
1900 데이터 수정
1901 [수정하기] button click
1902
1903
1904 28. All Codes
1905 1)HomeController.java
1906
1907 package com.example.biz;
1908
1909 import java.util.List;
1910
1911 import org.springframework.beans.factory.annotation.Autowired;
1912 import org.springframework.stereotype.Controller;
1913 import org.springframework.ui.Model;
1914 import org.springframework.web.bind.annotation.PathVariable;
1915 import org.springframework.web.bind.annotation.RequestMapping;
1916 import org.springframework.web.bind.annotation.RequestMethod;
1917 import org.springframework.web.bind.annotation.RequestParam;
1918
1919 import com.example.service.MemberService;
1920 import com.example.vo.MemberVO;
```

```
1921
1922 /**
1923  * Handles requests for the application home page.
1924  */
1925 @Controller
1926 public class HomeController {
1927     @Autowired
1928     MemberService memberService;
1929
1930     @RequestMapping(value = "/create", method = RequestMethod.POST)
1931     public String home(MemberVO memberVo, Model model) {
1932         int row = this.memberService.create(memberVo);
1933         if(row == 1) model.addAttribute("status", "Insert Success");
1934         else model.addAttribute("status", "Insert Failure");
1935         return "create";    // /WEB-INF/views/create.jsp
1936     }
1937
1938     @RequestMapping(value = "/view/{userid}", method = RequestMethod.GET)
1939     public String view(@PathVariable String userid, Model model) {
1940         MemberVO memberVo = this.memberService.read(userid);
1941         model.addAttribute("member", memberVo);
1942         return "view";
1943     }
1944
1945     @RequestMapping(value = "/list", method = RequestMethod.GET)
1946     public String list(Model model) {
1947         List<MemberVO> list = this.memberService.readAll();
1948         model.addAttribute("userlist", list);
1949         return "list";    // /WEB-INF/views/list.jsp
1950     }
1951
1952     @RequestMapping(value = "/delete/{userid}", method = RequestMethod.GET)
1953     public String delete(@PathVariable String userid) {
1954         this.memberService.delete(userid);
1955         return "redirect:/list";
1956     }
1957
1958     @RequestMapping(value = "/update", method = RequestMethod.POST)
1959     public String update(@RequestParam("userid") String userid,
1960         @RequestParam("age") int age,
1961         @RequestParam("gender") String gender,
1962         @RequestParam("city") String city) {
1963         this.memberService.update(
1964             new MemberVO(userid, "", age, gender, city));
1965         return "redirect:/list";
1966     }
1967 }
```

2)MemberServiceImpl.java

```
1970
1971 package com.example.service;
1972
1973 import java.util.List;
1974
1975 import org.springframework.beans.factory.annotation.Autowired;
1976 import org.springframework.stereotype.Service;
1977
1978 import com.example.dao.MemberDao;
```

```
1979     import com.example.vo.MemberVO;
1980
1981     @Service("memberService")
1982     public class MemberServiceImpl implements MemberService {
1983         @Autowired
1984         MemberDao memberDao;
1985
1986         @Override
1987         public int create(MemberVO memberVo) {
1988             return this.memberDao.create(memberVo);
1989         }
1990
1991         @Override
1992         public MemberVO read(String userid) {
1993             return this.memberDao.read(userid);
1994         }
1995
1996         @Override
1997         public List<MemberVO> readAll() {
1998             return this.memberDao.readAll();
1999         }
2000
2001         @Override
2002         public int update(MemberVO memberVo) {
2003             return this.memberDao.update(memberVo);
2004         }
2005
2006         @Override
2007         public int delete(String userid) {
2008             return this.memberDao.delete(userid);
2009         }
2010     }
2011
2012 3)MemberDaoImpl.java
2013
2014     package com.example.dao;
2015
2016     import java.sql.ResultSet;
2017     import java.sql.SQLException;
2018     import java.util.List;
2019
2020     import org.springframework.beans.factory.annotation.Autowired;
2021     import org.springframework.jdbc.core.JdbcTemplate;
2022     import org.springframework.jdbc.core.RowMapper;
2023     import org.springframework.stereotype.Repository;
2024
2025     import com.example.vo.MemberVO;
2026
2027     @Repository("memberDao")
2028     public class MemberDaoImpl implements MemberDao {
2029         @Autowired
2030         JdbcTemplate jdbcTemplate;
2031
2032         @Override
2033         public int create(MemberVO memberVo) {
2034             String sql = "INSERT INTO Member VALUES(?,?,?,?)";
2035             return this.jdbcTemplate.update(sql, memberVo.getUserid(),
                memberVo.getUsername(), memberVo.getAge(),
```

```

2036         memberVo.getGender(), memberVo.getCity());
2037     }
2038
2039     class MyRowMapper implements RowMapper<MemberVO> {
2040         @Override
2041         public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
2042             MemberVO memberVo = new MemberVO(rs.getString("userid"),
2043                 rs.getString("username"), rs.getInt("userage"),
2044                 rs.getString("gender"), rs.getString("city"));
2045             return memberVo;
2046         }
2047
2048         @Override
2049         public MemberVO read(String userid) {
2050             String sql = "SELECT * FROM Member WHERE userid = ?";
2051             return this.jdbcTemplate.queryForObject(sql, new Object[] { userid }, new
2052                 MyRowMapper());
2053
2054             @Override
2055             public List<MemberVO> readAll() {
2056                 String sql = "SELECT * FROM Member ORDER BY userid DESC";
2057                 return this.jdbcTemplate.query(sql, new MyRowMapper());
2058             }
2059
2060             @Override
2061             public int update(MemberVO memberVo) {
2062                 String sql = "UPDATE Member SET userage = ?, gender = ?, city = ? " + "WHERE
2063                     userid = ?";
2064                 return this.jdbcTemplate.update(sql, memberVo.getAge(), memberVo.getGender(),
2065                     memberVo.getCity(),
2066                     memberVo.getUserid());
2067             }
2068
2069             @Override
2070             public int delete(String userid) {
2071                 String sql = "DELETE FROM Member WHERE userid = ?";
2072                 return this.jdbcTemplate.update(sql, userid);
2073             }
2074         }
2075     }

```

Task7. Form Data Validation

1. Package Explorer > right-click > New > Other > Spring > Spring Legacy Project
2. Select Spring MVC Project
3. Project name : FormValidationDemo > Next
4. Enter a topLevelPackage : com.example.biz > Finish
5. pom.xml 수정하기


```

<properties>
    <java-version>1.8</java-version>
    <org.springframework-version>5.2.0.RELEASE</org.springframework-version>
    <org.aspectj-version>1.9.4</org.aspectj-version>
    <org.slf4j-version>1.7.28</org.slf4j-version>
</properties>
...
<dependency>

```

```
2090     <groupId>javax.servlet</groupId>
2091     <artifactId>javax.servlet-api</artifactId>
2092     <version>4.0.1</version>
2093     <scope>provided</scope>
2094 </dependency>
2095 <dependency>
2096     <groupId>javax.servlet.jsp</groupId>
2097     <artifactId>javax.servlet.jsp-api</artifactId>
2098     <version>2.3.3</version>
2099     <scope>provided</scope>
2100 </dependency>
2101 <dependency>
2102     <groupId>junit</groupId>
2103     <artifactId>junit</artifactId>
2104     <version>4.12</version>
2105     <scope>test</scope>
2106 </dependency>
2107
```

2108 6. pom.xml > right-click > Run As > Maven install
2109 [INFO] BUILD SUCCESS
2110

2111 7. FormValidationDemo Project > right-click > Properties > Project Facets > Select Java >
Change Version 1.8
2112 Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close
2113

2114 8. UserVO 객체 생성

2115 1)src/main/java/com.example.vo package 생성
2116 2)src/main/java/com.example.vo.UserVO class
2117

```
2118     package com.example.vo;
2119
2120     public class UserVO {
2121         private String name;
2122         private int age;
2123         private String userid;
2124         public String getName() {
2125             return name;
2126         }
2127         public void setName(String name) {
2128             this.name = name;
2129         }
2130         public int getAge() {
2131             return age;
2132         }
2133         public void setAge(int age) {
2134             this.age = age;
2135         }
2136         public String getUserid() {
2137             return userid;
2138         }
2139         public void setUserid(String userid) {
2140             this.userid = userid;
2141         }
2142         @Override
2143         public String toString() {
2144             return "UserVO [name=" + name + ", age=" + age + ", userid=" + userid + "];"
2145         }
2146     }
```

```
2147
2148
2149 9. Validator를 이용한 검증
2150 1)Data Command 객체에서 유효성 검사를 할 수 있다.
2151 2)UserValidator 객체 생성
2152 3)src/main/java/com.example.biz.UserValidator class
2153
2154     package com.example.biz;
2155
2156     import org.springframework.validation.Errors;
2157     import org.springframework.validation.Validator;
2158
2159     import com.example.vo.UserVO;
2160
2161     public class UserValidator implements Validator {
2162
2163         @Override
2164         public boolean supports(Class<?> arg0) {
2165             //검증할 객체의 class 타입 정보를 반환
2166             return UserVO.class.isAssignableFrom(arg0);
2167         }
2168
2169         @Override
2170         public void validate(Object obj, Errors errors) {
2171             System.out.println("검증시작");
2172             UserVO userVO = (UserVO)obj;
2173
2174             String username = userVO.getName();
2175             if(username == null || username.trim().isEmpty()) {
2176                 System.out.println("이름의 값이 빠졌습니다.");
2177                 errors.rejectValue("name", "No Value");
2178             }
2179
2180             int userage = userVO.getAge();
2181             if(userage == 0) {
2182                 System.out.println("나이의 값이 빠졌습니다.");
2183                 errors.rejectValue("age", "No Value");
2184             }
2185
2186             String userid = userVO.getUserid();
2187             if(userid == null || userid.trim().isEmpty()) {
2188                 System.out.println("아이디의 값이 빠졌습니다.");
2189                 errors.rejectValue("userid", "No Value");
2190             }
2191         }
2192     }
2193
2194 4)src/main/java/com.example.biz/HomeController.java
2195
2196     @RequestMapping(value = "/register", method=RequestMethod.GET)
2197     public String register() {
2198         return "register";
2199     }
2200
2201     @RequestMapping(value = "/register", method=RequestMethod.POST)
2202     public String register(@ModelAttribute("userVO") UserVO userVO, BindingResult result) {
2203         String page = "register_ok";
2204         UserValidator validator = new UserValidator();
```

```

2205     validator.validate(userVO, result);
2206     if(result.hasErrors()) {
2207         page = "register";
2208     }
2209     return page;
2210 }
2211
2212 5)src/main/webapp/WEB-INF/views/register.jsp
2213 <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
2214 <!DOCTYPE html>
2215 <html>
2216 <head>
2217 <meta charset="UTF-8">
2218 <title>회원 가입 폼</title>
2219 </head>
2220 <body>
2221     <form action="/biz/register" method="post">
2222         Name : <input type="text" name="name" /> <br />
2223         Age : <input type="number" name="age" /> <br />
2224         ID : <input type="text" name="userid" /> <br />
2225         <input type="submit" value="가입하기" />
2226     </form>
2227 </body>
2228 </html>
2229
2230 6)src/main/webapp/WEB-INF/views/register_ok.jsp
2231 <%@ page language="java" contentType="text/html; charset=UTF-8"
2232     pageEncoding="UTF-8"%>
2233 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2234 <c:set var="user" value="${userVO}" />
2235 <!DOCTYPE html>
2236 <html>
2237 <head>
2238 <meta charset="UTF-8">
2239 <title>회원 가입 결과 창</title>
2240 </head>
2241 <body>
2242     <ul>
2243         <li>이름 : ${user.name}</li>
2244         <li>나이 : ${user.age}</li>
2245         <li>아이디 : ${user.userid}</li>
2246     </ul>
2247 </body>
2248 </html>
2249
2250 7)Test
2251 http://localhost:8080/biz/register에서
2252 이름, 나이, 아이디를 모두 입력하면 결과창으로 넘어오고
2253 한 개라도 입력하지 않으면 다시 입력창으로 간다.
2254
2255 10. ValidationUtils class를 이용한 검증
2256 1)ValidationUtils class는 validate() method를 좀 더 편리하게 사용할 수 있게 해줌.
2257 2)UserValidator.java 수정
2258
2259     /*String username = userVO.getName();
2260     if(username == null || username.trim().isEmpty()) {
2261         System.out.println("이름의 값이 빠졌습니다.");

```



```

2262         errors.rejectValue("name", "No Value");
2263     }*/
2264
2265     ValidationUtils.rejectIfEmptyOrWhitespace(errors, "name", "No Value");
2266
2267 11. @Valid와 @InitBinder 이용하기
2268     1)Spring Framework이 대신 검증해 줌
2269     2)mvnrepository에서 'hibernate validator'로 검색
2270
2271     <dependency>
2272         <groupId>org.hibernate.validator</groupId>
2273         <artifactId>hibernate-validator</artifactId>
2274         <version>6.0.18.Final</version>
2275     </dependency>
2276
2277 3)pom.xml에 넣고 Maven Clean > Maven Install
2278 4)HomeController.java 수정
2279
2280     @RequestMapping(value = "/register", method=RequestMethod.POST)
2281     public String register(@ModelAttribute("userVO") @Valid UserVO userVO, BindingResult
2282     result) {
2283         String page = "register_ok";
2284         //UserValidator validator = new UserValidator();
2285         //validator.validate(userVO, result);
2286         if(result.hasErrors()) {
2287             page = "register";
2288         }
2289
2290         return page;
2291     }
2292
2293     @InitBinder
2294     protected void initBinder(WebDataBinder binder) {
2295         binder.setValidator(new UserValidator());
2296     }
2297
2298 12. Test
2299     http://localhost:8080/biz/register에서
2300     이름, 나이, 아이디를 모두 입력하면 결과창으로 넘어오고
2301     한 개라도 입력하지 않으면 다시 입력창으로 간다.
2302
2303 -----
2304 Task8. Convert J2EE to Spring MVC
2305 1. In J2EE Perspective
2306 2. Project Explorer > right-click > New > Dynamic Web Project
2307 3. Project name : SpringWebDemo > Next > Check [Generate web.xml deployment
2308 descriptor] > Finish
2309 4. Convert to Maven Project
2310     1)project right-click > Configure > Convert to Maven Project > Finish
2311     2)Project : /SpringWebDemo
2312     3)Group Id : SpringWebDemo
2313     4)Artifact Id : SpringWebDemo
2314     5)version : 0.0.1-SNAPSHOT
2315     6)Packaging : war
2316     7)Finish
2317 5. Add Spring Project Nature

```

```
2318 -project right-click > Spring Tools > Add Spring Project Nature
2319
2320 6. 새로 생성된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install
2321 <dependencies>
2322   <dependency>
2323     <groupId>org.springframework</groupId>
2324     <artifactId>spring-context</artifactId>
2325     <version>5.2.0.RELEASE</version>
2326   </dependency>
2327   <dependency>
2328     <groupId>junit</groupId>
2329     <artifactId>junit</artifactId>
2330     <version>4.12</version>
2331     <scope>test</scope>
2332   </dependency>
2333   <dependency>
2334     <groupId>org.springframework</groupId>
2335     <artifactId>spring-jdbc</artifactId>
2336     <version>5.2.0.RELEASE</version>
2337   </dependency>
2338 </dependencies>
2339
2340 7. Spring mvc library 검색 및 설치
2341 1) http://mvnrepository.com에서 'spring mvc'로 검색
2342 2) pom.xml에 추가
2343
2344   <dependency>
2345     <groupId>org.springframework</groupId>
2346     <artifactId>spring-webmvc</artifactId>
2347     <version>5.2.0.RELEASE</version>
2348   </dependency>
2349
2350 3) Maven Clean > Maven Install
2351
2352 8. Build path에 config folder 추가
2353 1) project right-click > Build Path > Configure Build Path > Select [Source] tab
2354 2) Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
2355 3) Folder name : config > Finish > OK > Apply and Close
2356 4) Java Resources > config 폴더 확인
2357
2358 9. config folder에 beans.xml file 생성
2359 1) Spring Perspective로 전환
2360 2) config > right-click > New > Other > Spring > Spring Bean Configuration File >
   beans.xml
2361 3) 생성시 beans, context, mvc 체크
2362 <?xml version="1.0" encoding="UTF-8"?>
2363 <beans xmlns="http://www.springframework.org/schema/beans"
2364   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2365   xmlns:context="http://www.springframework.org/schema/context"
2366   xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans.xsd
   http://www.springframework.org/schema/context
   http://www.springframework.org/schema/context/spring-context-3.2.xsd">
2367
2368
2369 </beans>
2370
2371 10. ContextLoaderListener class 설정
2372 1) web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener -
```

ContextLoaderListener] 를 선택하면 아래의 code가 자동 삽입

```

2373
2374 <!-- needed for ContextLoaderListener -->
2375 <context-param>
2376   <param-name>contextConfigLocation</param-name>
2377   <param-value>location</param-value>
2378 </context-param>
2379
2380 <!-- Bootstraps the root web application context before servlet initialization -->
2381 <listener>
2382   <listener-class>org.springframework.web.context.ContextLoaderListener</listener-clas
2383   s>
2384 </listener>

```

2)아래 code로 변환

```

2385 <context-param>
2386   <param-name>contextConfigLocation</param-name>
2387   <param-value>classpath:beans.xml</param-value>
2388 </context-param>
2389
2390

```

11. DispatcherServlet Class 추가

1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherservlet - DispatcherServlet declaration] 선택하면 아래의 code가 자동 추가된다.

```

2393
2394 <!-- The front controller of this Spring Web application, responsible for handling all
2395 application requests -->
2396 <servlet>
2397   <servlet-name>springDispatcherServlet</servlet-name>
2398   <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
2399   <init-param>
2400     <param-name>contextConfigLocation</param-name>
2401     <param-value>location</param-value>
2402   </init-param>
2403   <load-on-startup>1</load-on-startup>
2404 </servlet>
2405
2406 <!-- Map all requests to the DispatcherServlet for handling -->
2407 <servlet-mapping>
2408   <servlet-name>springDispatcherServlet</servlet-name>
2409   <url-pattern>url</url-pattern>
2410 </servlet-mapping>

```

2)아래의 code로 변환

```

2411 <init-param>
2412   <param-name>contextConfigLocation</param-name>
2413   <param-value>classpath:beans*.xml</param-value>
2414 </init-param>
2415
2416
2417 <servlet-mapping>
2418   <servlet-name>springDispatcherServlet</servlet-name>
2419   <url-pattern>*.do</url-pattern>
2420 </servlet-mapping>
2421

```

12. mvnrepository에서 'jstl'로 검색 후 설치

1)목록에서 2번째 : 1.2버전

```

2422
2423 <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
2424 <dependency>
2425
2426

```

```
2427         <groupId>javax.servlet</groupId>
2428         <artifactId>jstl</artifactId>
2429         <version>1.2</version>
2430     </dependency>
2431
2432     2)pom.xml에 붙여넣고 Maven Clean > Maven Install
2433
2434
2435 13. Hello Controller 작성
2436     1)src/com.example.vo package 생성
2437     2)src/com.example.vo.HelloVO class 생성
2438
2439     package com.example.vo;
2440
2441     public class HelloVO {
2442         private String name;
2443
2444         public void setName(String name) {
2445             this.name = name;
2446         }
2447
2448         public String sayHello() {
2449             return "Hello " + name;
2450         }
2451     }
2452
2453     3)src/com.example.controller package 생성
2454     4)com.example.controller.HelloController class 생성
2455
2456     package com.example.controller;
2457
2458     import org.springframework.beans.factory.annotation.Autowired;
2459     import org.springframework.stereotype.Controller;
2460     import org.springframework.ui.Model;
2461     import org.springframework.web.bind.annotation.RequestMapping;
2462
2463     import com.example.vo.HelloVO;
2464
2465     @Controller
2466     public class HelloController {
2467         @Autowired
2468         private HelloVO helloBean;
2469
2470         @RequestMapping("/hello.do")
2471         public String hello(Model model) {
2472             String msg = helloBean.sayHello();
2473             model.addAttribute("greet", msg);
2474             return "hello.jsp";
2475         }
2476     }
2477
2478
2479 14. beans.xml 수정
2480     <context:component-scan base-package="com.example" />
2481
2482     <bean id="helloVO" class="com.example.vo.HelloVO">
2483         <property name="name" value="한지민" />
2484     </bean>
```

```
2485
2486 15. WebContent/hello.jsp 생성
2487
2488 <%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
2489 <!DOCTYPE html>
2490 <html>
2491     <head>
2492         <meta charset="UTF-8">
2493         <title>Insert title here</title>
2494     </head>
2495     <body>
2496         ${greet}
2497     </body>
2498 </html>
2499
2500 16. project > right-click > Run As > Run on Server > Finish
2501
2502 17. http://localhost:8080/SpringWebDemo/hello.do
2503     Hello 한지민
2504
2505
2506 -----
2507 Task9. Convert J2EE to Spring MVC
2508 1. In J2EE Perspective
2509 2. Project Explorer > right-click > New > Dynamic Web Project
2510 3. Project name : SpringWebDemo1 > Next > Check [Generate web.xml deployment
    descriptor] > Finish
2511 4. Convert to Maven Project
2512     -project right-click > Configure > Convert to Maven Project > Finish
2513 5. Add Spring Project Nature
2514     -project right-click > Spring Tools > Add Spring Project Nature
2515
2516 6. 새로 생성된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install
2517 <dependencies>
2518     <dependency>
2519         <groupId>org.springframework</groupId>
2520         <artifactId>spring-context</artifactId>
2521         <version>5.2.0.RELEASE</version>
2522     </dependency>
2523     <dependency>
2524         <groupId>junit</groupId>
2525         <artifactId>junit</artifactId>
2526         <version>4.12</version>
2527         <scope>test</scope>
2528     </dependency>
2529     <dependency>
2530         <groupId>org.springframework</groupId>
2531         <artifactId>spring-jdbc</artifactId>
2532         <version>5.2.0.RELEASE</version>
2533     </dependency>
2534     <dependency>
2535         <groupId>javax.servlet</groupId>
2536         <artifactId>jstl</artifactId>
2537         <version>1.2</version>
2538     </dependency>
2539     <dependency>
2540         <groupId>com.oracle</groupId>
```

```

2541     <artifactId>ojdbc6</artifactId>
2542     <version>11.2</version>
2543 </dependency>
2544 <dependency>
2545     <groupId>org.springframework</groupId>
2546     <artifactId>spring-webmvc</artifactId>
2547     <version>5.2.0.RELEASE</version>
2548 </dependency>
2549 </dependencies>
2550
2551 2)Maven Clean > Maven Install
2552
2553
2554 7. Build path에 config folder 추가
2555 1)project right-click > Build Path > Configure Build Path > Select [Source] tab
2556 2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
2557 3)Folder name : config > Finish > OK > Apply and Close
2558 4)Java Resources > config 폴더 확인
2559
2560
2561 8. config folder에 beans.xml file 생성
2562 1)Spring Perspective로 전환
2563 2)config Folder > right-click > New > Spring Bean Configuration File
2564 3)File name : beans.xml
2565 4)생성시 beans,context, mvc 체크
2566 <?xml version="1.0" encoding="UTF-8"?>
2567 <beans xmlns="http://www.springframework.org/schema/beans"
2568     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2569     xmlns:context="http://www.springframework.org/schema/context"
2570     xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd">
2571
2572
2573
2574 </beans>
2575
2576
2577 9. ContextLoaderListener class 설정
2578 1)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener -
ContextLoaderListener] 를 선택하면 아래의 코드가 자동 삽입
2579
2580 <!-- needed for ContextLoaderListener -->
2581 <context-param>
2582     <param-name>contextConfigLocation</param-name>
2583     <param-value>location</param-value>
2584 </context-param>
2585
2586 <!-- Bootstraps the root web application context before servlet initialization -->
2587 <listener>
2588     <listener-class>org.springframework.web.context.ContextLoaderListener</listener-clas
s>
2589 </listener>
2590
2591 2)아래 코드로 변환
2592 <context-param>
2593     <param-name>contextConfigLocation</param-name>
2594     <param-value>classpath:beans.xml</param-value>

```

```

2595     </context-param>
2596
2597
2598 10. DispatcherServlet Class 추가
2599 1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherservlet -
    DispatcherServlet declaration] 선택하면 아래의 코드가 자동 추가된다.
2600
2601     <!-- The front controller of this Spring Web application, responsible for handling all
    application requests -->
2602     <servlet>
2603         <servlet-name>springDispatcherServlet</servlet-name>
2604         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
2605         <init-param>
2606             <param-name>contextConfigLocation</param-name>
2607             <param-value>location</param-value>
2608         </init-param>
2609         <load-on-startup>1</load-on-startup>
2610     </servlet>
2611
2612     <!-- Map all requests to the DispatcherServlet for handling -->
2613     <servlet-mapping>
2614         <servlet-name>springDispatcherServlet</servlet-name>
2615         <url-pattern>url</url-pattern>
2616     </servlet-mapping>
2617
2618 2)아래의 코드로 변환
2619     <init-param>
2620         <param-name>contextConfigLocation</param-name>
2621         <param-value>classpath:beans*.xml</param-value>
2622     </init-param>
2623
2624     <servlet-mapping>
2625         <servlet-name>springDispatcherServlet</servlet-name>
2626         <url-pattern>*.do</url-pattern>
2627     </servlet-mapping>
2628
2629
2630 11. UserVO class 생성
2631 1)src/com.example.vo package 생성
2632 2)src/com.example.vo.UserVO class 생성
2633
2634     package com.example.vo;
2635
2636     public class UserVO {
2637         private String userId;
2638         private String name;
2639         private String gender;
2640         private String city;
2641         public UserVO() {}
2642         public UserVO(String userId, String name, String gender, String city) {
2643             this.userId = userId;
2644             this.name = name;
2645             this.gender = gender;
2646             this.city = city;
2647         }
2648         public String getUserId() {
2649             return userId;
2650         }

```

```
2651     public void setId(String id) {
2652         this.id = id;
2653     }
2654     public String getName() {
2655         return name;
2656     }
2657     public void setName(String name) {
2658         this.name = name;
2659     }
2660     public String getGender() {
2661         return gender;
2662     }
2663     public void setGender(String gender) {
2664         this.gender = gender;
2665     }
2666     public String getCity() {
2667         return city;
2668     }
2669     public void setCity(String city) {
2670         this.city = city;
2671     }
2672     @Override
2673     public String toString() {
2674         return "UserVO [id=" + id + ", name=" + name + ", gender=" + gender +
2675             ", city=" + city + "]\n";
2676     }
2677 }
2678
```

12. UserDao 객체 생성

```
2680 1)src/com.example.dao package 생성
2681 2)src/com.example.dao.UserDao interface
2682
2683     package com.example.dao;
2684
2685     import java.util.List;
2686
2687     import com.example.vo.UserVO;
2688
2689     public interface UserDao {
2690         void insert(UserVO user);
2691
2692         List<UserVO> readAll();
2693
2694         void update(UserVO user);
2695
2696         void delete(String id);
2697
2698         UserVO read(String id);
2699     }
2700
2701 -src/com.example.dao.UserDaoImplJDBC.java 생성
2702
2703     package com.example.dao;
2704
2705     import java.sql.ResultSet;
2706     import java.sql.SQLException;
2707     import java.util.List;
```



```
2708
2709     import javax.sql.DataSource;
2710
2711     import org.springframework.beans.factory.annotation.Autowired;
2712     import org.springframework.dao.EmptyResultDataAccessException;
2713     import org.springframework.jdbc.core.JdbcTemplate;
2714     import org.springframework.jdbc.core.RowMapper;
2715     import org.springframework.stereotype.Repository;
2716
2717     import com.example.vo.UserVO;
2718
2719     @Repository("userDao")
2720     public class UserDaoImplJDBC implements UserDao {
2721
2722         private JdbcTemplate jdbcTemplate;
2723
2724         @Autowired
2725         public void setDataSource(DataSource dataSource) {
2726             this.jdbcTemplate = new JdbcTemplate(dataSource);
2727         }
2728
2729         class UserMapper implements RowMapper<UserVO> {
2730             public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
2731                 UserVO user = new UserVO();
2732                 user.setUserId(rs.getString("userId"));
2733                 user.setName(rs.getString("name"));
2734                 user.setGender(rs.getString("gender"));
2735                 user.setCity(rs.getString("city"));
2736                 return user;
2737             }
2738         }
2739
2740         @Override
2741         public void insert(UserVO user) {
2742             String SQL = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
2743             jdbcTemplate.update(SQL, user.getUserId(), user.getName(), user.getGender(),
2744                                 user.getCity());
2745
2746             System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
2747                                 user.getName());
2748         }
2749
2750         @Override
2751         public List<UserVO> readAll() {
2752             String SQL = "SELECT * FROM users";
2753             List<UserVO> userList = jdbcTemplate.query(SQL, new UserMapper());
2754             return userList;
2755         }
2756
2757         @Override
2758         public void update(UserVO user) {
2759             String SQL = "UPDATE users SET name = ?, gender = ?, city = ? WHERE userid =
2760                             ?";
2761             jdbcTemplate.update(SQL, user.getName(), user.getGender(), user.getCity(),
2762                                 user.getUserId());
2763             System.out.println("갱신된 Record with ID = " + user.getUserId());
2764         }
2765     }
```

```
2762     @Override
2763     public void delete(String id) {
2764         String SQL = "DELETE FROM users WHERE userid = ?";
2765         jdbcTemplate.update(SQL, id);
2766         System.out.println("삭제된 Record with ID = " + id);
2767     }
2768
2769     @Override
2770     public UserVO read(String id) {
2771         String SQL = "SELECT * FROM users WHERE userid = ?";
2772         try {
2773             UserVO user = jdbcTemplate.queryForObject(SQL, new Object[] { id }, new
                UserMapper());
2774             return user;
2775         } catch (EmptyResultDataAccessException e) {
2776             return null;
2777         }
2778     }
2779 }
```

2782 13. UserService 객체 생성

- 2783 1)src/com.example.service package 생성
- 2784 2)src/com.example.service.UserService interface

```
2785
2786     package com.example.service;
2787
2788     import java.util.List;
2789
2790     import com.example.vo.UserVO;
2791
2792     public interface UserService {
2793         void insertUser(UserVO user);
2794
2795         List<UserVO> getUserList();
2796
2797         void deleteUser(String id);
2798
2799         UserVO getUser(String id);
2800
2801         void updateUser(UserVO user);
2802     }
```

2804 3)src/com.example.service.UserServiceImpl.java

```
2805
2806     package com.example.service;
2807
2808     import java.util.List;
2809
2810     import org.springframework.beans.factory.annotation.Autowired;
2811     import org.springframework.stereotype.Service;
2812
2813     import com.example.dao.UserDao;
2814     import com.example.vo.UserVO;
2815
2816     @Service("userService")
2817     public class UserServiceImpl implements UserService {
2818         @Autowired
```

```
2819     UserDao userDao;
2820
2821     @Override
2822     public void insertUser(UserVO user) {
2823         this.userDao.insert(user);
2824     }
2825
2826     @Override
2827     public List<UserVO> getUserList() {
2828         return this.userDao.readAll();
2829     }
2830
2831     @Override
2832     public void deleteUser(String id) {
2833         this.userDao.delete(id);
2834     }
2835
2836     @Override
2837     public UserVO getUser(String id) {
2838         return this.userDao.read(id);
2839     }
2840
2841     @Override
2842     public void updateUser(UserVO user) {
2843         this.userDao.update(user);
2844     }
2845 }
```

2848 14. UserController 객체 생성

- 2849 1)src/com.example.controller package 생성
- 2850 2)com.example.controller.UserController class 생성

```
2851
2852     package com.example.controller;
2853
2854     import org.springframework.beans.factory.annotation.Autowired;
2855     import org.springframework.stereotype.Controller;
2856
2857     import com.example.service.UserService;
2858
2859     @Controller
2860     public class UserController {
2861         @Autowired
2862         private UserService userService;
2863
2864         @RequestMapping("/userInfo.do")
2865         public String getUserList(@RequestParam("userId") String userId, Model model) {
2866             UserVO user = userService.getUser(userId);
2867             //System.out.println(user);
2868             model.addAttribute("user", user);
2869             return "userInfo.jsp";
2870         }
2871     }
```

2874 15. config/dbinfo.properties file 생성

```
2875
2876     db.driverClass=oracle.jdbc.driver.OracleDriver
```

```

2877 db.url=jdbc:oracle:thin:@localhost:1521:XE
2878 db.username=hr
2879 db.password=hr
2880
2881
2882 16. beans.xml 수정
2883
2884 <context:component-scan base-package="com.example" />
2885
2886 <context:property-placeholder location="classpath:dbinfo.properties" />
2887 <bean id="dataSource"
2888     class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
2889     <property name="driverClass" value="${db.driverClass}" />
2890     <property name="url" value="${db.url}" />
2891     <property name="username" value="${db.username}" />
2892     <property name="password" value="${db.password}" />
2893 </bean>
2894
2895 17. WebContent/index.jsp 생성
2896
2897 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2898 <c:redirect url="userInfo.do" />
2899
2900
2901 18. WebContent/userInfo.jsp 생성
2902
2903 <%@ page language="java" contentType="text/html; charset=UTF-8"
2904     pageEncoding="UTF-8"%>
2905 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2906 <c:set var="user" value="${user}" />
2907 <!DOCTYPE html>
2908 <html>
2909 <head>
2910 <meta charset="UTF-8">
2911 <title>Insert title here</title>
2912 </head>
2913 <body>
2914 <h1>userInfo.jsp</h1>
2915 <h2>사용자 정보</h2>
2916 아이디 : ${user.userId} <br />
2917 이름 : ${user.name} <br />
2918 성별 : ${user.gender} <br />
2919 도시 : ${user.city} <br />
2920 </body>
2921 </html>
2922
2923 19. project > right-click > Run As > Run on Server > Finish
2924
2925 20. http://localhost:8080/SpringWebDemo/userinfo.do?userId=scott
2926
2927 -----
2928
2929 Task10. File Upload with Spring MVC
2930 1. In J2EE Perspective
2931 2. Project Explorer > right-click > New > Dynamic Web Project
2932 3. Project name : FileUploadDemo > Next > Check [Generate web.xml deployment

```

```

descriptor] > Finish
2933 4. Convert to Maven Project
2934 1)project right-click > Configure > Convert to Maven Project > Finish
2935
2936 5. Add Spring Project Nature
2937 1)project right-click > Spring Tools > Add Spring Project Nature
2938
2939 6. 새로 생성된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install
2940 <dependencies>
2941 <dependency>
2942 <groupId>org.springframework</groupId>
2943 <artifactId>spring-context</artifactId>
2944 <version>5.2.0.RELEASE</version>
2945 </dependency>
2946 <dependency>
2947 <groupId>org.springframework</groupId>
2948 <artifactId>spring-webmvc</artifactId>
2949 <version>5.2.0.RELEASE</version>
2950 </dependency>
2951 </dependencies>
2952
2953
2954 7. ContextLoaderListener class 설정
2955 1)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener -
ContextLoaderListener] 를 선택하면 아래의 코드가 자동 삽입
2956
2957 <!-- needed for ContextLoaderListener -->
2958 <context-param>
2959 <param-name>contextConfigLocation</param-name>
2960 <param-value>location</param-value>
2961 </context-param>
2962
2963 <!-- Bootstraps the root web application context before servlet initialization -->
2964 <listener>
2965 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-clas
s>
2966 </listener>
2967
2968 2)아래 코드로 변환
2969 <context-param>
2970 <param-name>contextConfigLocation</param-name>
2971 <param-value>classpath:applicationContext.xml</param-value>
2972 </context-param>
2973
2974
2975 8. DispatcherServlet Class 추가
2976 1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherservlet -
DispatcherServlet declaration] 선택하면 아래의 코드가 자동 추가된다.
2977
2978 <!-- The front controller of this Spring Web application, responsible for handling all
application requests -->
2979 <servlet>
2980 <servlet-name>springDispatcherServlet</servlet-name>
2981 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
2982 <init-param>
2983 <param-name>contextConfigLocation</param-name>
2984 <param-value>location</param-value>
2985 </init-param>

```

```
2986     <load-on-startup>1</load-on-startup>
2987 </servlet>
2988
2989 <!-- Map all requests to the DispatcherServlet for handling -->
2990 <servlet-mapping>
2991     <servlet-name>springDispatcherServlet</servlet-name>
2992     <url-pattern>url</url-pattern>
2993 </servlet-mapping>
2994
2995 2)아래의 코드로 변환
2996 <init-param>
2997     <param-name>contextConfigLocation</param-name>
2998     <param-value>classpath:beans.xml</param-value>
2999 </init-param>
3000
3001 <servlet-mapping>
3002     <servlet-name>springDispatcherServlet</servlet-name>
3003     <url-pattern>/</url-pattern>
3004 </servlet-mapping>
3005
3006
3007 9. FileUpload library 추가
3008 1)Apache에서 제공하는 Common FileUpload library를 사용하여 file upload를 처리하기 위한 library
3009 2)mvnrepository에서 'common fileupload'라고 검색하여 library 추가
3010 <dependency>
3011     <groupId>commons-fileupload</groupId>
3012     <artifactId>commons-fileupload</artifactId>
3013     <version>1.4</version>
3014 </dependency>
3015
3016 3)mvnrepository에서 'commons io'라고 검색하여 library 추가
3017 <dependency>
3018     <groupId>commons-io</groupId>
3019     <artifactId>commons-io</artifactId>
3020     <version>2.6</version>
3021 </dependency>
3022
3023 4)Maven Clean > Maven Install
3024
3025
3026 10. Thumbnail Image Library 추가
3027 1)mvnrepository에서 'imgscalr-lib'라고 검색하여 library 추가
3028 <dependency>
3029     <groupId>org.imgscalr</groupId>
3030     <artifactId>imgscalr-lib</artifactId>
3031     <version>4.2</version>
3032 </dependency>
3033
3034 2)Maven Clean > Maven Install
3035
3036
3037 11. Build path에 config Foler 추가
3038 1)project right-click > Build Path > Configure Build Path > Select [Source] tab
3039 2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
3040 3)Folder name : config > Finish > OK > Apply and Close
3041 4)Java Resources > config 폴더 확인
3042
3043
```

```
3044 12. config Folder에 applicationContext.xml file 생성
3045 1)config > right-click > New > Other > Spring > Spring Bean Configuration File
3046 2)Name : applicationContext.xml
3047 3)Namespace Tab에서 context, mvc check 할 것
3048
3049 <?xml version="1.0" encoding="UTF-8"?>
3050 <beans xmlns="http://www.springframework.org/schema/beans"
3051     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3052     xmlns:context="http://www.springframework.org/schema/context"
3053     xmlns:mvc="http://www.springframework.org/schema/mvc"
3054     xsi:schemaLocation="http://www.springframework.org/schema/mvc
3055         http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
3056         http://www.springframework.org/schema/beans
3057         http://www.springframework.org/schema/beans/spring-beans.xsd
3058         http://www.springframework.org/schema/context
3059         http://www.springframework.org/schema/context/spring-context-4.3.xsd">
3060
3061     <context:component-scan
3062         base-package="com.example" />
3063     <mvc:annotation-driven />
3064 </beans>
3065
3066 13. config Folder에 beans.xml file 생성
3067 1)config > right-click > New > Other > Spring > Spring Bean Configuration File
3068 2)Name : beans.xml
3069 3)beans.xml에 Spring multipartResolver 추가
3070
3071 <bean id="multipartResolver"
3072     class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
3073     <property name="maxUploadSize" value="10240000" />
3074     <property name="defaultEncoding" value="utf-8" />
3075 </bean>
3076
3077 14. web.xml에 한글 File Encoding 처리하기
3078 1)한글 File이 Upload될 때 File 명이 깨지는 것을 해결하기 위해 web.xml에 아래 내용을 추가한다.
3079
3080 <filter>
3081     <filter-name>encodingFilter</filter-name>
3082     <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
3083     <init-param>
3084         <param-name>encoding</param-name>
3085         <param-value>UTF-8</param-value>
3086     </init-param>
3087     <init-param>
3088         <param-name>forceEncoding</param-name>
3089         <param-value>true</param-value>
3090     </init-param>
3091 </filter>
3092 <filter-mapping>
3093     <filter-name>encodingFilter</filter-name>
3094     <url-pattern>/*</url-pattern>
3095 </filter-mapping>
3096
3097 15. View 작성
3098 1)WebContent/form.jsp
```

```

3098
3099     <%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
3100     <!DOCTYPE html>
3101     <html>
3102     <head>
3103     <meta charset="UTF-8">
3104     <title>Insert title here</title>
3105     </head>
3106     <body>
3107         <h1>file 업로드 예제</h1>
3108         <form method="post" action="upload" enctype="multipart/form-data">
3109             <label>email:</label> <input type="text" name="email"> <br>
3110             <br> <label>file:</label> <input type="file" name="file1">
3111             <br>
3112             <br> <input type="submit" value="upload">
3113         </form>
3114     </body>
3115 </html>
3116
3117
3118 16. Service 작성
3119     1)src/com.example.service package
3120     2)src/com.example.service.FileUploadService.java
3121
3122     package com.example.service;
3123
3124     import java.io.FileOutputStream;
3125     import java.io.IOException;
3126     import java.util.Calendar;
3127
3128     import org.springframework.stereotype.Service;
3129     import org.springframework.web.multipart.MultipartFile;
3130
3131     @Service
3132     public class FileUploadService {
3133         // 리눅스 기준으로 file 경로를 작성 ( 루트 경로인 /으로 시작한다. )
3134         // 윈도우라면 workspace의 드라이브를 파악하여 JVM이 알아서 처리해준다.
3135         // 따라서 workspace가 C드라이브에 있다면 C드라이브에 upload 폴더를 생성해 놓아야 한다.
3136         private static final String SAVE_PATH = "/upload";
3137         private static final String PREFIX_URL = "/upload/";
3138
3139         public String restore(MultipartFile multipartFile) {
3140             String uri = null;
3141
3142             try {
3143                 // file 정보
3144                 String originFilename = multipartFile.getOriginalFilename();
3145                 String extName = originFilename.substring(originFilename.lastIndexOf("."),
originFilename.length());
3146                 Long size = multipartFile.getSize();
3147
3148                 // 서버에서 저장 할 file 이름
3149                 String saveFileName = genSaveFileName(extName);
3150
3151                 System.out.println("originFilename : " + originFilename);
3152                 System.out.println("extensionName : " + extName);
3153                 System.out.println("size : " + size);

```



```

3154         System.out.println("saveFileName : " + saveFileName);
3155
3156         writeFile(multipartFile, saveFileName);
3157         uri = PREFIX_URL + saveFileName;
3158     }
3159     catch (IOException e) {
3160         // 원래라면 RuntimeException 을 상속받은 예외가 처리되어야 하지만
3161         // 편의상 RuntimeException을 던진다.
3162         // throw new FileUploadException();
3163         throw new RuntimeException(e);
3164     }
3165     return uri;
3166 }
3167
3168 // 현재 시간을 기준으로 file 이름 생성
3169 private String genSaveFileName(String extName) {
3170     String fileName = "";
3171
3172     Calendar calendar = Calendar.getInstance();
3173     fileName += calendar.get(Calendar.YEAR);
3174     fileName += calendar.get(Calendar.MONTH);
3175     fileName += calendar.get(Calendar.DATE);
3176     fileName += calendar.get(Calendar.HOUR);
3177     fileName += calendar.get(Calendar.MINUTE);
3178     fileName += calendar.get(Calendar.SECOND);
3179     fileName += calendar.get(Calendar.MILLISECOND);
3180     fileName += extName;
3181
3182     return fileName;
3183 }
3184
3185 // file을 실제로 write 하는 메서드
3186 private boolean writeFile(MultipartFile multipartFile, String saveFileName)
3187     throws IOException{
3188     boolean result = false;
3189
3190     byte[] data = multipartFile.getBytes();
3191     FileOutputStream fos = new FileOutputStream(SAVE_PATH + "/" + saveFileName);
3192     fos.write(data);
3193     fos.close();
3194
3195     return result;
3196 }
3197 }
3198 }
3199 }
3200

```

3)SAVE_PATH는 file을 저장할 위치를 가리킨다.

-일반적으로 server는 Linux 기반이므로 Linux 경로명을 사용하는 것이 좋다.

-즉 file을 root 경로인 / 아래의 upload folder에 저장하겠다는 의미인데, Windows에서는 JVM이 알아서 workspace가 존재하는 drive의 위치를 찾아서 drive를 root 경로로 하여 upload folder에 저장한다.

-예를들어 Eclipse workspace가 C drive에 있다면 C drive의 upload folder에 file이 저장될 것이다.

4)PREFIX_URL은 저장된 file을 JSP에서 불러오기 위한 경로를 의미한다.

5)MultipartFile 객체는 file의 정보를 담고 있다.

6)uri을 반환하는 이유는 view page에서 바로 image file을 보기 위함이다.

-만약 DB에서 image 경로를 저장 해야 한다면, 이와 같이 uri을 반환하면 좋을 것이다.

7)현재 시간을 기준으로 file 이름을 바꾼다.

-이렇게 하는 이유는, 여러 사용자가 올린 file의 이름이 같을 경우 덮어 씌어지는 문제가 발생하기 때문이다.

3211 -따라서 file 이름이 중복될 수 있는 문제를 해결하기 위해 ms단위의 시스템 시간을 이용하여 file 이름을 변경한다.
3212 8)FileOutputStream 객체를 이용하여 file을 저장한다.
3213
3214
3215 17. Controller 작성
3216 1)com.example.controller package
3217 2)com.example.controller.FileUploadController.java
3218
3219 package com.example.controller;
3220
3221 import org.springframework.beans.factory.annotation.Autowired;
3222 import org.springframework.stereotype.Controller;
3223 import org.springframework.ui.Model;
3224 import org.springframework.web.bind.annotation.RequestMapping;
3225 import org.springframework.web.bind.annotation.RequestMethod;
3226 import org.springframework.web.bind.annotation.RequestParam;
3227 import org.springframework.web.multipart.MultipartFile;
3228
3229 import com.example.service.FileUploadService;
3230
3231 @Controller
3232 public class FileUploadController {
3233 @Autowired
3234 FileUploadService fileUploadService;
3235
3236 @RequestMapping("/form")
3237 public String form() {
3238 return "form.jsp";
3239 }
3240
3241 @RequestMapping(value = "/upload", method = RequestMethod.POST)
3242 public String upload(@RequestParam("email") String email, @RequestParam("file1")
3243 MultipartFile file, Model model) {
3244 String uri = fileUploadService.restore(file);
3245 model.addAttribute("uri", uri);
3246 return "result.jsp";
3247 }
3248 }
3249
3250 18. WebContent/result.jsp
3251 <%@ page language="java" contentType="text/html; charset=UTF-8"
3252 pageEncoding="UTF-8"%>
3253 <!DOCTYPE html>
3254 <html>
3255 <head>
3256 <meta charset="UTF-8">
3257 <title>Insert title here</title>
3258 </head>
3259 <body>
3260 <h1>Upload completed</h1>
3261 <div class="result-images">
3262
3263 </div>
3264 <p>
3265 다시 업로드 하기
3266 </p>
</body>

```

3267     </html>
3268
3269
3270 19. C:/(현재 workspace가 C:라면)upload Folder 생성할 것
3271
3272 20. Project > right-click > Run As > Run on Server
3273     http://localhost:8080/FileUploadDemo/form
3274
3275
3276 21. 문제점 및 해결
3277     1)Upload Folder(C:/upload)를 보면 File이 Upload된 것을 확인할 수 있지만, 결과 화면을 보면 Image가 제
3278     대로 출력 되지 않을 것이다.
3279     2)Image File을 right-click하여 경로를 보면 아마 다음과 같을 것이다.
3280     3)http://localhost:8080/FileUploadDemo/upload/업로드한 파일
3281     4)File을 저장할 때 [upload]라는 Folder에 저장을 했는데, File을 저장할 때의 Upload는 C Drive 내의
3282     [upload] Folder이고,
3283     5)위 URL에서 [upload]는 Application 상 경로에 있는 upload이므로 WEB-INF 폴더의 하위 folder로서의
3284     upload를 의미한다.
3285     6)즉 실제 File이 저장된 Server 상의 위치( 물리 주소 )와 Application에서 보여주고자 하는 File 경로( 가상 주소
3286     )가 일치하지 않은 것이다.
3287     7)따라서 실제 File이 저장되어 있는 위치와 Application 상의 위치를 일치시키는 작업이 필요하다.
3288     8)beans.xml에 물리 주소와 가상 주소를 mapping 해주는 code를 추가하도록 해야한다.
3289
3290     <!-- resource mapping -->
3291     <!-- location : 물리적 주소 / mapping : 가상 주소 -->
3292     <mvc:resources location="file:///C:/upload/" mapping="/upload/*"/>
3293
3294     9)이제 정상적으로 result.jsp에서 image가 출력될 것이다.
3295
3296
3297 22. Multiple File Upload
3298     1)이번에는 여러 개의 File을 Upload 할 수 있는 Multiple Upload를 알아보자.
3299     2)수정할 부분은 <input> tag와 Controller에서 MultipartFile 객체를 받는 Parameter 부분 두 곳인데, 필요
3300     한 부분만 보자.
3301     3)form.jsp
3302
3303     <input type="file" name="files" multiple>
3304
3305     4)<input> 태그에서는 multiple 속성만 추가하면 된다.
3306     5)"File선택"을 클릭하면 ctrl 키를 눌러서 여러 개의 File을 선택할 수 있다.
3307
3308     6)FileUploadController
3309
3310     @RequestMapping( "/upload" )
3311     public String upload(@RequestParam String email,
3312     @RequestParam(required=false) List<MultipartFile> files, Model model) {
3313
3314     ...
3315
3316     }
3317
3318     7)Controller에서는 여러 개의 File을 받기 때문에 MultipartFile을 List로 받아야 한다.

```