

```
1  HOL : Spring MVC
2  -----
3  Task1. Spring MVC Demo
4  1. Package Explorer > right-click > New > Other > Spring > Spring Legacy Project
5  2. Select Spring MVC Project
6  3. Project name : HelloWorldWeb > Next
7  4. Enter a topLevelPackage : com.example.biz > Finish
8  5. pom.xml 수정하기
9      <properties>
10         <java-version>1.8</java-version>
11         <org.springframework-version>5.2.0.RELEASE</org.springframework-version>
12         <org.aspectj-version>1.9.4</org.aspectj-version>
13         <org.slf4j-version>1.7.28</org.slf4j-version>
14     </properties>
15     ...
16     <dependency>
17         <groupId>javax.servlet</groupId>
18         <artifactId>javax.servlet-api</artifactId>
19         <version>4.0.1</version>
20         <scope>provided</scope>
21     </dependency>
22     <dependency>
23         <groupId>javax.servlet.jsp</groupId>
24         <artifactId>javax.servlet.jsp-api</artifactId>
25         <version>2.3.3</version>
26         <scope>provided</scope>
27     </dependency>
28     <dependency>
29         <groupId>junit</groupId>
30         <artifactId>junit</artifactId>
31         <version>4.12</version>
32         <scope>test</scope>
33     </dependency>
34
35 6. pom.xml > right-click > Run As > Maven install
36 [INFO] BUILD SUCCESS
37
38 7. HelloWorldWeb project > right-click > Properties > Project Facets > Select Java > Change Version 1.8
39 -Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close
40
41 8. HelloWorldWeb Project right-click > Run As > Run on Server > Finish
42
43 9. http://localhost:8080/biz/
44
45 Hello world!
46
47 The time on the server is 2019년 6월 11일 (화) 오후 11시 40분 58초.<--원래 한글 깨짐
48
49 10. 한글 깨짐을 수정하는 것은 src/main/webapp/WEB-INF/views/home.jsp에서
50 <%@ page session="false" pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>로 수정
51
52 11. Context name 변경하기
```

53 1)Package Explorer에서 Servers/Tomcat v9.0 Server at localhost-config/server.xml에서 다음과 같이 수정
한다.

54 -path="/biz" --> path="/demo"

55 <Context docBase="HelloWorldWeb" path="/demo" reloadable="true"
source="org.eclipse.jst.jee.server:HelloWorldWeb"/>

56

57 2)수정 후 restart 하면 <http://localhost:8080/biz> --> <http://localhost:8080/demo> 로 변경됨

58

59

60 -----

61 Task2. resources Folder 이용하기

62 1. Image 경로 알아내기

63 1)src/main/webapp/resources/에 images Folder를 STS Package Explorer에서 생성한다.

64 2)Download받은 image를 src/main/webapp/resources/images/에 넣는다.

65 3)home.jsp에 아래 code를 추가한다.

66 <p></p>

67 4)Image가 잘 나온다.

68

69

70 2. Image 경로 변경

71 1)apple.jpg image 경로를 src/main/webapp/images/로 이동.

72 2)하지만 이렇게 하면 image가 보이지 않는다.

73 3)왜냐하면, servlet-context.xml에서 resource의 경로는 <resources mapping="/resources/**"
location="/resources/" /> 이기 때문.

74 4)즉, 기본적으로 resources folder 아래에서 resource를 찾는다.

75

76

77 3. <resources /> 추가

78 1)다시 resources Folder 하위로 images Folder를 복사

79 2)/src/main/webapp/하위에 images Folder를 생성하고 image를 넣고 home.jsp에 아래의 code를 추가한
다.

80 <p></p>

81 <p></p>

82 3)하지만 아래의 image는 보이지 않는다.

83 4)왜냐하면 새로 추가한 images Folder는 servlet-context.xml에서 설정하지 않았기 때문.

84 5)Image를 보이게 하기 위해 servlet-context.xml에 아래의 Code를 추가한다.

85 <resources mapping="/resources/**" location="/resources/" />

86 <resources mapping="/images/**" location="/images/" />

87 6)src/main/webapp/images Folder 추가

88 7)Project right-click > Run As > Run on Server > Restart >

89 -Image가 제대로 2개가 나온다.

90

91

92 -----

93 Task3. Controller Class 제작하기

94 1. 제작순서

95 1)@Controller를 이용한 class 생성

96 2)@RequestMapping을 이용한 요청 경로 지정

97 3)요청 처리 method 구현

98 4)View 이름 return

99

100 5)src/main/java/com.example.biz.UserController class 생성

```
101
102     @Controller
103     public class UserController {
104
105
106 2. 요청 처리 method 생성
107
108     package com.example.biz;
109
110     import org.springframework.stereotype.Controller;
111     import org.springframework.ui.Model;
112     import org.springframework.web.bind.annotation.RequestMapping;
113     import org.springframework.web.bind.annotation.RequestMethod;
114     import org.springframework.web.servlet.ModelAndView;
115
116     @Controller
117     public class UserController {
118         @RequestMapping("/view")
119         public String view(Model model){
120             /*
121             model.addAttribute("username", "한지민");
122             model.addAttribute("userage", 24);
123             model.addAttribute("job", "Developer");
124             return "view";
125             */
126             model.addAttribute("currentDate", new java.util.Date());
127             return "view";    // /WEB-INF/views/view + .jsp
128         }
129
130         @RequestMapping("/fruits")
131         public String fruits(Model model){
132             String [] array = {"Apple", "Mango", "Lemon", "Grape"};
133
134             model.addAttribute("fruits", array);
135
136             return "fruits";    // /WEB-INF/views/fruits + .jsp
137         }
138     }
139
140
141 3. View에 Data 전달
142 1)src/main/webapp/WEB-INF/views/view.jsp 생성
143
144     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
145     <!DOCTYPE html>
146     <html>
147         <head>
148             <meta charset="UTF-8">
149             <title>Insert title here</title>
150         </head>
151         <body>
152             <h1>view.jsp 입니다.</h1>
```

```

153     현재 날짜와 시간은 ${currentDate} 입니다.
154     </body>
155     </html>
156
157 2)src/main/webapp/WEB-INF/views/fruits.jsp 생성
158
159     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
160     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
161     <!DOCTYPE html>
162     <html>
163     <head>
164     <meta charset="UTF-8">
165     <title>Insert title here</title>
166     </head>
167     <body>
168     <h2>fruits.jsp</h2>
169     <ul>과일 종류
170     <c:forEach items="${fruits}" var="fruit">
171     <li>${fruit}</li>
172     </c:forEach>
173     </ul>
174     </body>
175     </html>
176
177 3)http://localhost:8080/demo/view --> /view.jsp
178 4)http://localhost:8080/demo/fruits --> /fruits.jsp
179
180
181 4. View에 ModelAndView 객체로 data 전달
182 1)UserController.java에 아래의 코드 추가
183
184     @RequestMapping(value = "/demo", method = RequestMethod.GET)
185     public ModelAndView demo() {
186         /*
187         ModelAndView mav = new ModelAndView("view2");
188         mav.addObject("username", "한지민");
189         mav.addObject("currentDate", new java.util.Date());
190         return mav;
191         */
192         ModelAndView mav = new ModelAndView();
193         mav.addObject("userid", "example");
194         mav.addObject("passwd", "12345678");
195         mav.setViewName("/demo");
196         return mav;
197     }
198
199 2)src/main/webapp/WEB-INF/views/demo.jsp 생성
200
201     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
202     <!DOCTYPE html">
203     <html>
204     <head>

```

```

205     <meta charset="UTF-8">
206     <title>Insert title here</title>
207 </head>
208 <body>
209     아이디 : ${userid} <br />
210     패스워드 : ${passwd}
211 </body>
212 </html>

```

```

213
214 3) http://localhost:8080/demo/demo --> /demo.jsp

```

```

215     아이디 : example
216     패스워드 : 12345678

```

```

217

```

```

218

```

```

219 5. Controller class에 @RequestMapping 적용

```

```

220 1)src/main/java/com.example.biz.StudentController.java 생성

```

```

221

```

```

222     package com.example.biz;

```

```

223

```

```

224     import org.springframework.stereotype.Controller;

```

```

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

```

```

226     import org.springframework.web.bind.annotation.RequestMethod;

```

```

227     import org.springframework.web.servlet.ModelAndView;

```

```

228

```

```

229     @Controller

```

```

230     @RequestMapping("/bbs")

```

```

231     public class StudentController {

```

```

232

```

```

233         @RequestMapping(value="/get", method = RequestMethod.GET)

```

```

234         public ModelAndView getStudent() {

```

```

235

```

```

236             ModelAndView mav = new ModelAndView();

```

```

237             mav.setViewName("/bbs/get"); // /WEB-INF/views/bbs/get.jsp

```

```

238             mav.addObject("name", "한지민");

```

```

239             mav.addObject("age", 25);

```

```

240             return mav;

```

```

241         }

```

```

242     }

```

```

243

```

```

244 2)src/main/webapp/WEB-INF/views/bbs/get.jsp

```

```

245     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

```

```

246     <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

```

```

247     "http://www.w3.org/TR/html4/loose.dtd">

```

```

248

```

```

249     <html>

```

```

250     <head>

```

```

251     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

```

```

252     <title>Insert title here</title>

```

```

253     </head>

```

```

254     <body>

```

```

255         학생 이름 : ${name} <br />

```

```

256         학생 나이 : ${age}

```

```

257     </body>

```

```
256     </html>
257
258 3) http://localhost:8080/demo/bbs/get
259     학생 이름 : 한지민
260     학생 나이 : 25
261
262
263 -----
264 Task4. 다양한 GET Request 처리하기
265 1. Package Explorer > right-click > New > Spring Legacy Project
266 2. Select Spring MVC Project
267 3. Project name : MVCDemo > Next
268 4. Enter a topLevelPackage : com.example.biz > Finish
269 5. pom.xml 수정하기
270     <properties>
271         <java-version>1.8</java-version>
272         <org.springframework-version>5.2.0.RELEASE</org.springframework-version>
273         <org.aspectj-version>1.9.4</org.aspectj-version>
274         <org.slf4j-version>1.7.28</org.slf4j-version>
275     </properties>
276     ...
277     <dependency>
278         <groupId>javax.servlet</groupId>
279         <artifactId>javax.servlet-api</artifactId>
280         <version>4.0.1</version>
281         <scope>provided</scope>
282     </dependency>
283     <dependency>
284         <groupId>javax.servlet.jsp</groupId>
285         <artifactId>javax.servlet.jsp-api</artifactId>
286         <version>2.3.3</version>
287         <scope>provided</scope>
288     </dependency>
289     <dependency>
290         <groupId>junit</groupId>
291         <artifactId>junit</artifactId>
292         <version>4.12</version>
293         <scope>test</scope>
294     </dependency>
295
296 6. pom.xml > right-click > Run As > Maven install
297     [INFO] BUILD SUCCESS
298
299 7. Project > right-click > Properties > Project Facets > Select Java > Change Version 1.8
300     -Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close
301
302 8. src/main/java/com.example.biz/RequestController.java 생성
303
304     package com.example.biz;
305
306     import org.springframework.stereotype.Controller;
307
```

```
308 @Controller
309 public class RequestController {
310
311 9. HttpServletRequest class 이용하기
312 1)RequestController.java
313
314 @RequestMapping(value="/confirm", method=RequestMethod.GET)
315 public String confirm(HttpServletRequest request, Model model) {
316     String userid = request.getParameter("userid");
317     String passwd = request.getParameter("passwd");
318     String name = request.getParameter("name");
319     int age = Integer.parseInt(request.getParameter("age"));
320     String gender = request.getParameter("gender");
321
322     model.addAttribute("userid", userid);
323     model.addAttribute("passwd", passwd);
324     model.addAttribute("name", name);
325     model.addAttribute("age", age);
326     model.addAttribute("gender", gender);
327     return "confirm"; // /WEB-INF/views/confirm.jsp
328 }
329
330 2)src/main/webapp/WEB-INF/views/confirm.jsp
331
332 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
333 <!DOCTYPE html>
334 <html>
335     <head>
336         <meta charset="UTF-8">
337         <title>Insert title here</title>
338     </head>
339     <body>
340         아이디 : ${userid} <br />
341         패스워드 : ${passwd} <br />
342         사용자 이름 : ${name} <br />
343         나이 : ${age} <br />
344         성별 : ${gender} <br />
345     </body>
346 </html>
347
348 3)Project right-click > Run As > Run on Server > restart
349 4)localhost:8080/biz/confirm?name=한지민&gender=여성&age=25&userid=jimin&passwd=1234
350     아이디 : jimin
351     패스워드 : 1234
352     사용자 이름 : 한지민
353     나이 : 25
354     성별 : 여성
355
356
357 10. @RequestParam Annotation 이용하기
358 1)RequestController.java
359 @RequestMapping(value="/confirm", method=RequestMethod.GET)
```

```

360     public String confirm(@RequestParam("userid") String userid,
361                           @RequestParam("passwd") String passwd,
362                           @RequestParam("name") String name,
363                           @RequestParam("age") int age,
364                           @RequestParam("gender") String gender ,Model model) {
365
366         model.addAttribute("userid", userid);
367         model.addAttribute("passwd", passwd);
368         model.addAttribute("name", name);
369         model.addAttribute("age", age);
370         model.addAttribute("gender", gender);
371         return "confirm"; // /WEB-INF/views/confirm.jsp
372     }
373
374 2)src/main/webapp/WEB-INF/views/confirm.jsp
375
376 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
377 <!DOCTYPE html>
378 <html>
379     <head>
380         <meta charset="UTF-8">
381         <title>Insert title here</title>
382     </head>
383     <body>
384         아이디 : ${userid} <br />
385         패스워드 : ${passwd} <br />
386         사용자 이름 : ${name} <br />
387         나이 : ${age} <br />
388         성별 : ${gender} <br />
389     </body>
390 </html>
391
392 3)localhost:8080/biz/confirm?name=한지민&gender=여성&age=25&userid=jimin&passwd=1234
393 아이디 : jimin
394 패스워드 : 1234
395 사용자 이름 : 한지민
396 나이 : 25
397 성별 : 여성
398
399
400 11. Data Commander 객체 사용하기1
401 1)src/main/java/com.example.vo.UserVO.java 생성
402
403 package com.example.vo;
404
405 public class UserVO {
406     private String userid;
407     private String passwd;
408     private String name;
409     private int age;
410     private String gender;
411

```



```
412     public UserVO(){
413     public UserVO(String userid, String passwd, String name, int age, String gender){
414         this.userid = userid;
415         this.passwd = passwd;
416         this.name = name;
417         this.age = age;
418         this.gender = gender;
419     }
420     public String getUserid() {
421         return userid;
422     }
423     public void setUserid(String userid) {
424         this.userid = userid;
425     }
426     public String getPasswd() {
427         return passwd;
428     }
429     public void setPasswd(String passwd) {
430         this.passwd = passwd;
431     }
432     public String getName() {
433         return name;
434     }
435     public void setName(String name) {
436         this.name = name;
437     }
438     public int getAge() {
439         return age;
440     }
441     public void setAge(int age) {
442         this.age = age;
443     }
444     public String getGender() {
445         return gender;
446     }
447     public void setGender(String gender) {
448         this.gender = gender;
449     }
450     @Override
451     public String toString() {
452         return "UserVO [userid=" + userid + ", passwd=" + passwd + ", name=" + name + ", age=" +
453             + gender + "];";
454     }
455 }
456
457 2)RequestController.java
458
459 @RequestMapping(value="/confirm", method=RequestMethod.GET)
460 public String confirm(@RequestParam("userid") String userid,
461     @RequestParam("passwd") String passwd,
462     @RequestParam("name") String name,
```

```

463         @RequestParam("age") int age,
464         @RequestParam("gender") String gender ,Model model) {
465
466         UserVO userVO = new UserVO();
467         userVO.setUserid(userid);
468         userVO.setPasswd(passwd);
469         userVO.setName(name);
470         userVO.setAge(age);
471         userVO.setGender(gender);
472
473         model.addAttribute("userVO", userVO);
474
475         return "confirm1"; // /WEB-INF/views/confirm1.jsp
476     }
477
478 3)src/main/webapp/WEB-INF/views/confirm1.jsp
479
480     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
481     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
482     <c:set var="user" value="${userVO}"/>
483     <!DOCTYPE html>
484     <html>
485     <head>
486     <meta charset="UTF-8">
487     <title>Insert title here</title>
488     </head>
489     <body>
490         <h1>confirm1.jsp</h1>
491         <h2>사용자 정보</h2>
492         아이디 : ${user.userid} <br />
493         패스워드 : ${user.passwd} <br />
494         이름 : ${user.name} <br />
495         나이 : ${user.age} <br />
496         성별 : ${user.gender}
497     </body>
498     </html>
499
500 4)localhost:8080/biz/confirm?name=한지민&gender=여성&age=25&userid=jimin&passwd=1234
501     confirm1.jsp
502
503     사용자 정보
504
505     아이디 : jimin
506     패스워드 : 1234
507     사용자 이름 : 한지민
508     나이 : 25
509     성별 : 여성
510
511
512 12. Data Commander 객체 이용하기2
513     1)RequestController.java
514

```

```

515     @RequestMapping(value="/confirm", method=RequestMethod.GET)
516     public String confirm(UserVO userVO) {
517
518         return "confirm2"; // /WEB-INF/views/confirm2.jsp
519     }
520
521 2)src/main/webapp/WEB-INF/views/confirm2.jsp
522
523     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
524     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
525     <c:set var="user" value="${userVO}"/>
526     <!DOCTYPE html>
527     <html>
528     <head>
529     <meta charset="UTF-8">
530     <title>Insert title here</title>
531     </head>
532     <body>
533         <h1>confirm2.jsp</h1>
534         <h2>사용자 정보</h2>
535         아이디 : ${user.userid} <br />
536         비밀번호 : ${user.passwd} <br />
537         이름 : ${user.name} <br />
538         나이 : ${user.age} <br />
539         성별 : ${user.gender}
540     </body>
541     </html>
542
543 3)localhost:8080/biz/confirm?name=한지민&gender=여성&age=25&userid=jimin&passwd=1234
544     confirm2.jsp
545
546     사용자 정보
547
548     아이디 : jimin
549     비밀번호 : 1234
550     사용자 이름 : 한지민
551     나이 : 25
552     성별 : 여성
553
554
555 13. @PathVariable 이용하기
556 1)RequestController.java
557
558     @RequestMapping(value="/confirm/{userid}/{passwd}/{name}/{age}/{gender}",
559     method=RequestMethod.GET)
560     public String confirm(@PathVariable String userid, @PathVariable String passwd,
561     @PathVariable String name, @PathVariable int age,
562     @PathVariable String gender,Model model) {
563         model.addAttribute("userInfo", new UserVO(userid, passwd, name, age, gender));
564         return "confirm3";
565     }

```

```

566 2)src/main/webapp/WEB-INF/views/confirm3.jsp
567
568 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
569 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
570 <c:set var="user" value="${userInfo}"/>
571 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
572 <html>
573 <head>
574 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
575 <title>Insert title here</title>
576 </head>
577 <body>
578 <h1>confirm3.jsp</h1>
579 <h2>사용자 정보</h2>
580 아이디 : ${user.userid} <br />
581 비밀번호 : ${user.passwd} <br />
582 이름 : ${user.name} <br />
583 나이 : ${user.age} <br />
584 성별 : ${user.gender}
585 </body>
586 </html>
587
588 3)localhost:8080/biz/confirm/jimin/1234/한지민/25/여성
589 confirm3.jsp
590
591 사용자 정보
592
593 아이디 : jimin
594 비밀번호 : 1234
595 사용자 이름 : 한지민
596 나이 : 25
597 성별 : 여성
598
599
600 -----
601 Task5. @RequestMapping Parameter 다루기
602 1. GET 방식과 POST 방식
603 1)src/main/java/com.example.biz/HomeController.java
604
605 @RequestMapping(value="/login", method=RequestMethod.POST)
606 public String login(@RequestParam("userid") String userid,
607                    @RequestParam("passwd") String passwd,
608                    Model model) {
609
610     model.addAttribute("userid", userid);
611     model.addAttribute("passwd", passwd);
612     return "login";
613 }
614
615 2)src/main/webapp/resources/login.html
616 <!DOCTYPE html>

```

```

617 <html>
618 <head>
619 <meta charset="UTF-8">
620 <title>로그인 폼</title>
621 </head>
622 <body>
623 <form method="GET" action="/biz/login">
624 아이디 : <input type="text" name="userid" /> <br />
625 패스워드 : <input type="password" name="passwd" /> <br />
626 <input type="submit" value="로그인하기" />
627 </form>
628 </body>
629 </html>

```

- 630
- 631 3) <http://localhost:8080/biz/resources/login.html>에서 submit 하면 405 error 발생
- 632 4) 왜냐하면 서로의 method가 불일치하기 때문
- 633 5) 해결방법
- 634 -src/main/java/com.example.biz/HomeController.java 수정
- 635 -즉 login method(요청 처리 method)의 이름은 같지만 parameter의 type과 return type이 틀리기 때문
에 Method Overloading 됨.

```

636
637 @RequestMapping(value="/login", method=RequestMethod.POST)
638 public String login(@RequestParam("userid") String userid,
639                    @RequestParam("passwd") String passwd,
640                    Model model) {
641
642     model.addAttribute("userid", userid);
643     model.addAttribute("passwd", passwd);
644     return "login";
645 }
646 @RequestMapping(value="/login", method=RequestMethod.GET)
647 public ModelAndView login(@RequestParam("userid") String userid,
648                          @RequestParam("passwd") String passwd) {
649
650     ModelAndView mav = new ModelAndView();
651     mav.addObject("userid", userid);
652     mav.addObject("passwd", passwd);
653     mav.setViewName("login");
654     return mav;
655 }

```

656

657 6)src/main/webapp/WEB-INF/views/login.jsp

658

```

659 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
660 <!DOCTYPE html>
661 <html>
662 <head>
663 <meta charset="UTF-8">
664 <title>Insert title here</title>
665 </head>
666 <body>
667 아이디 : ${userid} <br />

```

```

668     패스워드 : ${passwd}
669 </body>
670 </html>
671
672 7) http://localhost:8080/biz/resources/login.html
673 아이디 : jimin
674 패스워드 : 1234
675
676
677 2. @ModelAttribute Annotation 이용하기
678 1)@ModelAttribute Annotation을 이용하면 Data Commander 객체의 이름을 변경할 수 있다.
679 2)src/main/webapp/resources/register.html
680
681 <!DOCTYPE html>
682 <html>
683 <head>
684 <meta charset="UTF-8">
685 <title>회원가입 폼</title>
686 </head>
687 <body>
688 <form method="POST" action="/biz/register">
689   아이디 : <input type="text" name="userid" /> <br />
690   패스워드 : <input type="password" name="passwd" /> <br />
691   이름 : <input type="text" name="name" /> <br />
692   나이 : <input type="number" name="age" /> <br />
693   성별 : <input type="radio" name="gender" value="남성" /> 남성 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
694         <input type="radio" name="gender" value="여성" /> 여성 <br />
695   <input type="submit" value="가입하기" />
696 </form>
697 </body>
698 </html>
699
700 3)src/main/java/com.example.biz/HomeController.java
701
702 @RequestMapping(value="/register", method=RequestMethod.POST)
703 public String register(@ModelAttribute("u") UserVO userVO) {    //userVO가 아니라 u로 변경
704
705     return "register";
706 }
707
708 4)src/main/webapp/WEB-INF/views/register.jsp
709
710 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
711 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
712 <c:set var="user" value="${u}" />
713 <!DOCTYPE html>
714 <html>
715 <head>
716 <meta charset="UTF-8">
717 <title>Insert title here</title>
718 </head>
719 <body>

```

```

720     <h1>사용자 정보</h1>
721     <ul>
722         <li>아이디 : ${user.userid}</li>
723         <li>패스워드 : ${user.passwd}</li>
724         <li>이름 : ${user.name}</li>
725         <li>나이 : ${user.age}</li>
726         <li>성별 : ${user.gender}</li>
727     </ul>
728 </body>
729 </html>
730
731 5)Spring에서 POST 방식으로 Data를 보낼 때 한글깨짐 현상 발생
732 6)해결방법
733 7)web.xml
734
735     <filter>
736         <filter-name>encodingFilter</filter-name>
737         <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
738         <init-param>
739             <param-name>encoding</param-name>
740             <param-value>UTF-8</param-value>
741         </init-param>
742     </filter>
743     <filter-mapping>
744         <filter-name>encodingFilter</filter-name>
745         <url-pattern>/*</url-pattern>
746     </filter-mapping>
747
748 8)http://localhost:8080/biz/resources/register.html -->
749 9)http://localhost:8080/biz/register
750     사용자 정보
751
752     아이디 : jimin
753     패스워드 : 1234
754     사용자 이름 : 한지민
755     나이 : 25
756     성별 : 여성
757
758
759 3. redirect: 키워드 이용하기
760 1)src/main/java/com.example.biz/HomeController.java
761
762     @RequestMapping("/verify")
763     public String verify(HttpServletRequest request, Model model) {
764         String userid = request.getParameter("userid");
765         if(userid.equals("admin")) {    //만일 userid가 admin 이면 /admin으로 리다이렉트
766             return "redirect:admin";
767         }
768         return "redirect:user";        //만일 userid가 admin 이 아니면 /user로 리다이렉트
769         //return "redirect:http://www.naver.com";    //절대 경로도 가능
770     }
771

```

```

772     @RequestMapping("/admin")
773     public String verify1(Model model) {
774         model.addAttribute("authority", "관리자권한");
775         return "admin";
776     }
777
778     @RequestMapping("/user")
779     public String verify2(Model model) {
780         model.addAttribute("authority", "일반사용자");
781         return "user";
782     }
783
784 2)/src/main/webapp/WEB-INF/views/admin.jsp
785     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
786     <!DOCTYPE html>
787     <html>
788     <head>
789     <meta charset=UTF-8">
790     <title>Insert title here</title>
791     </head>
792     <body>
793         <h1>관리자 페이지</h1>
794         권한 : ${authority}
795     </body>
796     </html>
797
798 3)/src/main/webapp/WEB-INF/views/user.jsp
799
800     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
801     <!DOCTYPE html>
802     <html>
803     <head>
804     <meta charset=UTF-8">
805     <title>Insert title here</title>
806     </head>
807     <body>
808         <h1>일반 사용자 페이지</h1>
809         권한 : ${authority}
810     </body>
811     </html>
812
813 4)http://localhost:8080/biz/verify?userid=admin --> http://localhost:8080/biz/admin
814 5)http://localhost:8080/biz/verify?userid=user --> https://www.naver.com
815
816
817 -----
818 Task6. Database와 연동하기
819 1. Package Explorer > right-click > New > Spring Legacy Project
820 2. Select Spring MVC Project
821 3. Project name : MVCDemo1 > Next
822 4. Enter a topLevelPackage : com.example.biz > Finish
823 5. pom.xml 수정하기

```



```

824 <properties>
825   <java-version>1.8</java-version>
826   <org.springframework-version>5.2.0.RELEASE</org.springframework-version>
827   <org.aspectj-version>1.9.4</org.aspectj-version>
828   <org.slf4j-version>1.7.28</org.slf4j-version>
829 </properties>
830 ...
831 <dependency>
832   <groupId>javax.servlet</groupId>
833   <artifactId>javax.servlet-api</artifactId>
834   <version>4.0.1</version>
835   <scope>provided</scope>
836 </dependency>
837 <dependency>
838   <groupId>javax.servlet.jsp</groupId>
839   <artifactId>javax.servlet.jsp-api</artifactId>
840   <version>2.3.3</version>
841   <scope>provided</scope>
842 </dependency>
843 <dependency>
844   <groupId>junit</groupId>
845   <artifactId>junit</artifactId>
846   <version>4.12</version>
847   <scope>test</scope>
848 </dependency>
849
850
851 6. pom.xml > right-click > Run As > Maven install
852 [INFO] BUILD SUCCESS
853
854 7. MVCDemo1 Project > right-click > Properties > Project Facets > Select Java > Change Version 1.8
855 Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close
856
857
858 8. Create Table in MariaDB
859 CREATE TABLE Member
860 (
861   userid      VARCHAR(20),
862   username    VARCHAR(20) NOT NULL,
863   usage       TINYINT NOT NULL,
864   gender      VARCHAR(10) NOT NULL,
865   city        VARCHAR(50),
866   CONSTRAINT member_userid_pk PRIMARY KEY(userid)
867 );
868 -반드시 [test] Database의 조합을 utf8_general_ci로 맞출 것
869 -반드시 Member Table의 기본조합이 utf8_general_ci 임을 확인할 것
870
871
872 9. src/main/webapp/static folder 생성
873 1)src/main/webapp/static/css folder
874 2)src/main/webapp/static/images folder
875 3)src/main/webapp/static/js folder

```

```

876 -jquery-1.12.4.js
877 4)src/main/webapp/static/register.html
878 <!DOCTYPE html>
879 <html lang="en">
880 <head>
881   <meta charset="UTF-8">
882   <title>회원 가입</title>
883 </head>
884 <body>
885   <h1>회원 가입 창</h1>
886   <form action="/biz/create" method="post">
887     <ul>
888       <li>ID : <input type="text" name="userid" /> </li>
889       <li>이름 : <input type="text" name="username" /> </li>
890       <li>나이 : <input type="number" name="age" /> </li>
891       <li>성별 : <input type="radio" name="gender" value="남성"/> 남성
892               <input type="radio" name="gender" value="여성"/> 여성 </li>
893       <li>거주지 : <input type="text" name="city" /> </li>
894       <li><input type="submit" value="가입하기" /> </li>
895     </ul>
896   </form>
897 </body>
898 </html>
899
900
901 10. src/main/webapp/WEB-INF/spring/appServlet/sevlet-context.xml 수정
902   <resources mapping="/static/**" location="/static/" /> 추가
903
904   <context:component-scan base-package="com.example" /> 수정
905
906
907 11. src/main/resources/mariadb.properties
908 db.driverClass=org.mariadb.jdbc.Driver
909 db.url=jdbc:mariadb://localhost:3306/test
910 db.username=root
911 db.password=javamariadb
912
913
914 12. Spring JDBC 설치
915   1)JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
916
917   <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
918   <dependency>
919     <groupId>org.springframework</groupId>
920     <artifactId>spring-jdbc</artifactId>
921     <version>5.2.0.RELEASE</version>
922   </dependency>
923
924   2)pom.xml에 붙여 넣고 Maven Install 하기
925   [INFO] BUILD SUCCESS
926
927

```

```

928 13. MariaDB Jdbc Driver library 검색 및 설치
929 1)Maven Repository 에서 'mariadb'로 검색하여 MariaDB Java Client를 설치한다.
930
931 <dependency>
932     <groupId>org.mariadb.jdbc</groupId>
933     <artifactId>mariadb-java-client</artifactId>
934     <version>2.5.1</version>
935 </dependency>
936
937 2)pom.xml에 붙여 넣고 Maven Install 하기
938 [INFO] BUILD SUCCESS
939
940
941 14. src/main/webapp/WEB-INF/spring/root-context.xml
942 <?xml version="1.0" encoding="UTF-8"?>
943 <beans xmlns="http://www.springframework.org/schema/beans"
944     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
945     xmlns:context="http://www.springframework.org/schema/context"
946     xsi:schemaLocation="http://www.springframework.org/schema/beans
947         http://www.springframework.org/schema/beans/spring-beans.xsd
948         http://www.springframework.org/schema/context
949         http://www.springframework.org/schema/context/spring-context-4.3.xsd">
950
951 <!-- Root Context: defines shared resources visible to all other web components -->
952 <context:property-placeholder location="classpath:mariadb.properties"/>
953 <bean id="dataSource" class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
954     <property name="driverClass" value="${db.driverClass}" />
955     <property name="url" value="${db.url}" />
956     <property name="username" value="${db.username}" />
957     <property name="password" value="${db.password}" />
958 </bean>
959
960 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
961     <property name="dataSource" ref="dataSource" />
962 </bean>
963 </beans>
964
965 15. src/test/java/com.example.biz/TestApp class 생성
966 1)com.example.biz > right-click > New > JUnit Test Case
967 2)Select [New JUnit 4 test]
968 3)Name : TestApp
969 4)Finish
970 package com.example.biz;
971
972 import org.junit.Before;
973 import org.junit.Test;
974 import org.springframework.context.ApplicationContext;
975 import org.springframework.context.support.GenericXmlApplicationContext;
976 import org.springframework.jdbc.core.JdbcTemplate;
977
978 public class TestApp {

```

```
978     private ApplicationContext ctx;
979
980     @Before
981     public void init() {
982         this.ctx = new
983             GenericXmlApplicationContext("file:src/main/webapp/WEB-INF/spring**/root-context.xml");
984     }
985     @Test
986     public void test() {
987         JdbcTemplate jdbcTemplate = this.ctx.getBean("jdbcTemplate", JdbcTemplate.class);
988         System.out.println(jdbcTemplate);
989     }
990 }
```

991 5)Run as > JUnit Test > Green bar

992

993

994 16. package 생성

995 1)src/main/java/com.example.vo

996 2)src/main/java/com.example.dao

997 3)src/main/java/com.example.service

998

999

1000 17. src/com.example.vo.MemberVO class 생성

```
1001
1002     package com.example.vo;
1003
1004     public class MemberVO {
1005         private String userid;
1006         private String username;
1007         private int age;
1008         private String gender;
1009         private String city;
1010
1011         public MemberVO() {}
1012
1013         public MemberVO(String userid, String username, int age, String gender, String city) {
1014             this.userid = userid;
1015             this.username = username;
1016             this.age = age;
1017             this.gender = gender;
1018             this.city = city;
1019         }
1020
1021         public String getUserid() {
1022             return userid;
1023         }
1024
1025         public void setUserid(String userid) {
1026             this.userid = userid;
1027         }
1028     }
```

```
1029     public String getUsername() {
1030         return username;
1031     }
1032
1033     public void setUsername(String username) {
1034         this.username = username;
1035     }
1036
1037     public int getAge() {
1038         return age;
1039     }
1040
1041     public void setAge(int age) {
1042         this.age = age;
1043     }
1044
1045     public String getGender() {
1046         return gender;
1047     }
1048
1049     public void setGender(String gender) {
1050         this.gender = gender;
1051     }
1052
1053     public String getCity() {
1054         return city;
1055     }
1056
1057     public void setCity(String city) {
1058         this.city = city;
1059     }
1060
1061     @Override
1062     public String toString() {
1063         return "MemberVO [userid=" + userid + ", username=" + username + ", age=" + age + ",
1064             gender=" + gender
1065             + ", city=" + city + "];"
1066     }
1067 }
1068 18. com/example.dao
1069 1)MemberDao interface
1070     package com.example.dao;
1071
1072     import java.util.List;
1073
1074     import com.example.vo.MemberVO;
1075
1076     public interface MemberDao {
1077         int create(MemberVO memberVo);
1078         MemberVO read(String userid);
1079         List<MemberVO> readAll();
```

```
1080         int update(MemberVO memberVo);
1081         int delete(String userid);
1082     }
1083
1084 2)MemberDaoImpl.java
1085     package com.example.dao;
1086
1087     import java.util.List;
1088
1089     import org.springframework.beans.factory.annotation.Autowired;
1090     import org.springframework.jdbc.core.JdbcTemplate;
1091     import org.springframework.stereotype.Repository;
1092
1093     import com.example.vo.MemberVO;
1094
1095     @Repository("memberDao")
1096     public class MemberDaoImpl implements MemberDao {
1097         @Autowired
1098         JdbcTemplate jdbcTemplate;
1099
1100         @Override
1101         public int create(MemberVO memberVo) {
1102             return 0;
1103         }
1104
1105         @Override
1106         public MemberVO read(String userid) {
1107             return null;
1108         }
1109
1110         @Override
1111         public List<MemberVO> readAll() {
1112             return null;
1113         }
1114
1115         @Override
1116         public int update(MemberVO memberVo) {
1117             return 0;
1118         }
1119
1120         @Override
1121         public int delete(String userid) {
1122             return 0;
1123         }
1124     }
1125
1126 19. com.example.service
1127 1)MemberService interface
1128     package com.example.service;
1129
1130     import java.util.List;
1131
```

```
1132     import com.example.vo.MemberVO;
1133
1134     public interface MemberService {
1135         int create(MemberVO memberVo);
1136         MemberVO read(String userid);
1137         List<MemberVO> readAll();
1138         int update(MemberVO memberVo);
1139         int delete(String userid);
1140     }
1141
1142 2)MemberServiceImpl.java
1143     package com.example.service;
1144
1145     import java.util.List;
1146
1147     import org.springframework.beans.factory.annotation.Autowired;
1148     import org.springframework.stereotype.Service;
1149
1150     import com.example.dao.MemberDao;
1151     import com.example.vo.MemberVO;
1152
1153     @Service("memberService")
1154     public class MemberServiceImpl implements MemberService {
1155         @Autowired
1156         MemberDao memberDao;
1157
1158         @Override
1159         public int create(MemberVO memberVo) {
1160             return 0;
1161         }
1162
1163         @Override
1164         public MemberVO read(String userid) {
1165             return null;
1166         }
1167
1168         @Override
1169         public List<MemberVO> readAll() {
1170             return null;
1171         }
1172
1173         @Override
1174         public int update(MemberVO memberVo) {
1175             return 0;
1176         }
1177
1178         @Override
1179         public int delete(String userid) {
1180             return 0;
1181         }
1182     }
1183
```

```
1184
1185 20. com.example.biz
1186 1)HomeController.java
1187
1188     package com.example.biz;
1189
1190     import org.springframework.beans.factory.annotation.Autowired;
1191     import org.springframework.stereotype.Controller;
1192
1193     import com.example.service.MemberService;
1194
1195     /**
1196     * Handles requests for the application home page.
1197     */
1198     @Controller
1199     public class HomeController {
1200         @Autowired
1201         MemberService memberService;
1202     }
1203
1204
1205 21. Data Insert
1206 1)/src/main/webapp/static/register.html
1207 2)com.example.biz/HomeController.java
1208
1209     @Controller
1210     public class HomeController {
1211         @Autowired
1212         MemberService memberService;
1213
1214         @RequestMapping(value = "/create", method = RequestMethod.POST)
1215         public String home(MemberVO memberVo, Model model) {
1216             int row = this.memberService.create(memberVo);
1217             if(row == 1) model.addAttribute("status", "Insert Success");
1218             else model.addAttribute("status", "Insert Failure");
1219             return "create";    // /WEB-INF/views/create.jsp
1220         }
1221     }
1222
1223 3)com.example.service/MemberServiceImpl.java
1224
1225     @Service("memberService")
1226     public class MemberServiceImpl implements MemberService {
1227         @Autowired
1228         MemberDao memberDao;
1229
1230         @Override
1231         public int create(MemberVO memberVo) {
1232             return this.memberDao.create(memberVo);
1233         }
1234     }
1235
```


1236 4)com.example.dao.MemberDaoImpl.java

```
1237
1238 @Repository("memberDao")
1239 public class MemberDaoImpl implements MemberDao {
1240     @Autowired
1241     JdbcTemplate jdbcTemplate;
1242
1243     @Override
1244     public int create(MemberVO memberVo) {
1245         String sql = "INSERT INTO Member VALUES(?,?,?,?)";
1246         return this.jdbcTemplate.update(sql, memberVo.getUserid(),
1247             memberVo.getUsername(), memberVo.getAge(),
1248             memberVo.getGender(), memberVo.getCity());
1249     }
1250 }
```

1251

1252 5)views/create.jsp

```
1253
1254 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
1255 <!DOCTYPE html>
1256 <html>
1257 <head>
1258 <meta charset="UTF-8">
1259 <title>Insert title here</title>
1260 </head>
1261 <body>
1262     <h1>${status}</h1>
1263 </body>
1264 </html>
1265
```

1266 22. POST 발송시 한글 깨짐 처리하기

1267 1)web.xml

```
1268
1269 <filter>
1270     <filter-name>encodingFilter</filter-name>
1271     <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
1272     <init-param>
1273         <param-name>encoding</param-name>
1274         <param-value>UTF-8</param-value>
1275     </init-param>
1276 </filter>
1277 <filter-mapping>
1278     <filter-name>encodingFilter</filter-name>
1279     <url-pattern>/*</url-pattern>
1280 </filter-mapping>
1281
```

1282 23. Test

1283 1)<http://localhost:8080/biz/static/register.html>

1284 2)<http://localhost:8080/biz/create>

1285 Insert Success

1286

1287

```
1288 24. Data Select
1289 1)HomeController.java
1290
1291     @RequestMapping(value = "/view/{userid}", method = RequestMethod.GET)
1292     public String view(@PathVariable String userid, Model model) {
1293         MemberVO memberVo = this.memberService.read(userid);
1294         model.addAttribute("member", memberVo);
1295         return "view";
1296     }
1297
1298 2)MemberServiceImpl.java
1299
1300     @Override
1301     public MemberVO read(String userid) {
1302         return this.memberDao.read(userid);
1303     }
1304
1305 3)MemberDaoImpl.java
1306
1307     @Override
1308     public MemberVO read(String userid) {
1309         String sql = "SELECT * FROM Member WHERE userid = ?";
1310         return this.jdbcTemplate.queryForObject(sql, new Object[] {userid},
1311             new MyRowMapper());
1312     }
1313
1314     class MyRowMapper implements RowMapper<MemberVO>{
1315         @Override
1316         public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1317             MemberVO memberVo = new MemberVO(rs.getString("userid"),
1318                 rs.getString("username"), rs.getInt("userage"),
1319                 rs.getString("gender"), rs.getString("city"));
1320             return memberVo;
1321         }
1322     }
1323
1324 4)views/view.jsp
1325
1326     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
1327     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
1328     <c:set var="user" value="${member}" />
1329     <!DOCTYPE html>
1330     <html>
1331     <head>
1332         <meta charset="UTF-8">
1333         <title>Insert title here</title>
1334         <script src="/biz/static/js/jquery-1.12.4.js"></script>
1335         <script>
1336             $(function(){
1337                 $("#btnList").bind("click", function(){
1338                     location.href = "/biz/list";
1339                 });
```

```
1371 5)Test
1372 http://localhost:8080/biz/view/jimin
```

```
1376 1)HomeController.java
```

1385 2)MemberServiceImpl.java

Page 27 of 57

1392 3)MemberDaoImpl.java

```
1393
1394 @Override
1395 public List<MemberVO> readAll() {
1396     String sql = "SELECT * FROM Member ORDER BY userid DESC";
1397     return this.jdbcTemplate.query(sql, new MyRowMapper());
1398 }
```

1399
1400 4)iews/list.jsp

```
1401
1402 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
1403 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
1404 <!DOCTYPE html>
1405 <html>
1406 <head>
1407 <meta charset="UTF-8">
1408 <title>Insert title here</title>
1409 </head>
1410 <body>
1411 <h1>Member List</h1>
1412 <table border='1'>
1413 <thead>
1414 <tr>
1415 <th>아이디</th> <th>이름</th> <th>나이</th> <th>성별</th> <th>거주지</th>
1416 </tr>
1417 </thead>
1418 <tbody>
1419 <c:forEach items="${userlist}" var="user">
1420 <tr>
1421 <td> <a
1422 href="/biz/view/${user.userid}"> ${user.userid}</a> </td> <td> ${user.username}</td>
1423 <td> ${user.age}</td> <td> ${user.gender }</td>
1424 <td> ${user.city }</td>
1425 </tr>
1426 </c:forEach>
1427 </tbody>
1428 </table>
1429 </body>
1430 </html>
```

1431 5)Test

1432 <http://localhost:8080/biz/list>

1433

1434

1435 26. Data Delete

1436 1)HomeController.java

```
1437
1438 @RequestMapping(value = "/delete/{userid}", method = RequestMethod.GET)
1439 public String delete(@PathVariable String userid) {
1440     this.memberService.delete(userid);
1441     return "redirect:/list";
1442 }
```

```
1443
1444 2)MemberServiceImpl.java
1445
1446     @Override
1447     public int delete(String userid) {
1448         return this.memberDao.delete(userid);
1449     }
1450
1451 3)MemberDaoImpl.java
1452
1453     @Override
1454     public int delete(String userid) {
1455         String sql = "DELETE FROM Member WHERE userid = ?";
1456         return this.jdbcTemplate.update(sql, userid);
1457     }
1458
1459 4)Test
1460     http://localhost:8080/biz/delete/chulsu
1461
1462
1463 27. Data Update
1464 1)HomeController.java
1465
1466     @RequestMapping(value = "/update", method = RequestMethod.POST)
1467     public String update(@RequestParam("userid") String userid,
1468         @RequestParam("age") int age,
1469         @RequestParam("gender") String gender,
1470         @RequestParam("city") String city) {
1471         this.memberService.update(
1472             new MemberVO(userid, "", age, gender, city));
1473         return "redirect:/list";
1474     }
1475
1476 2)MemberServiceImpl.java
1477
1478     @Override
1479     public int update(MemberVO memberVo) {
1480         return this.memberDao.update(memberVo);
1481     }
1482
1483 3)MemberDaoImpl.java
1484
1485     @Override
1486     public int update(MemberVO memberVo) {
1487         String sql = "UPDATE Member SET usage = ?, gender = ?, city = ? " +
1488             "WHERE userid = ?";
1489         return this.jdbcTemplate.update(sql, memberVo.getAge(),
1490             memberVo.getGender(), memberVo.getCity(), memberVo.getUserid());
1491     }
1492
1493 4)Test
1494     http://localhost:8080/biz/list에서
```

```
1495     해당 ID Click
1496     데이터 수정
1497     [수정하기] button click
1498
1499
1500 28. All Codes
1501     1)HomeController.java
1502
1503     package com.example.biz;
1504
1505     import java.util.List;
1506
1507     import org.springframework.beans.factory.annotation.Autowired;
1508     import org.springframework.stereotype.Controller;
1509     import org.springframework.ui.Model;
1510     import org.springframework.web.bind.annotation.PathVariable;
1511     import org.springframework.web.bind.annotation.RequestMapping;
1512     import org.springframework.web.bind.annotation.RequestMethod;
1513     import org.springframework.web.bind.annotation.RequestParam;
1514
1515     import com.example.service.MemberService;
1516     import com.example.vo.MemberVO;
1517
1518     /**
1519     * Handles requests for the application home page.
1520     */
1521     @Controller
1522     public class HomeController {
1523         @Autowired
1524         MemberService memberService;
1525
1526         @RequestMapping(value = "/create", method = RequestMethod.POST)
1527         public String home(MemberVO memberVo, Model model) {
1528             int row = this.memberService.create(memberVo);
1529             if(row == 1) model.addAttribute("status", "Insert Success");
1530             else model.addAttribute("status", "Insert Failure");
1531             return "create";    // /WEB-INF/views/create.jsp
1532         }
1533
1534         @RequestMapping(value = "/view/{userid}", method = RequestMethod.GET)
1535         public String view(@PathVariable String userid, Model model) {
1536             MemberVO memberVo = this.memberService.read(userid);
1537             model.addAttribute("member", memberVo);
1538             return "view";
1539         }
1540
1541         @RequestMapping(value = "/list", method = RequestMethod.GET)
1542         public String list(Model model) {
1543             List<MemberVO> list = this.memberService.readAll();
1544             model.addAttribute("userlist", list);
1545             return "list";    // /WEB-INF/views/list.jsp
1546         }
1547     }
```

```
1547
1548     @RequestMapping(value = "/delete/{userid}", method = RequestMethod.GET)
1549     public String delete(@PathVariable String userid) {
1550         this.memberService.delete(userid);
1551         return "redirect:/list";
1552     }
1553
1554     @RequestMapping(value = "/update", method = RequestMethod.POST)
1555     public String update(@RequestParam("userid") String userid,
1556         @RequestParam("age") int age,
1557         @RequestParam("gender") String gender,
1558         @RequestParam("city") String city) {
1559         this.memberService.update(
1560             new MemberVO(userid, "", age, gender, city));
1561         return "redirect:/list";
1562     }
1563 }
```

2) MemberServiceImpl.java

```
1566     package com.example.service;
1567
1568     import java.util.List;
1569
1570     import org.springframework.beans.factory.annotation.Autowired;
1571     import org.springframework.stereotype.Service;
1572
1573     import com.example.dao.MemberDao;
1574     import com.example.vo.MemberVO;
1575
1576     @Service("memberService")
1577     public class MemberServiceImpl implements MemberService {
1578         @Autowired
1579         MemberDao memberDao;
1580
1581         @Override
1582         public int create(MemberVO memberVo) {
1583             return this.memberDao.create(memberVo);
1584         }
1585
1586         @Override
1587         public MemberVO read(String userid) {
1588             return this.memberDao.read(userid);
1589         }
1590
1591         @Override
1592         public List<MemberVO> readAll() {
1593             return this.memberDao.readAll();
1594         }
1595
1596         @Override
1597         public int update(MemberVO memberVo) {
```

```
1599         return this.memberDao.update(memberVo);
1600     }
1601
1602     @Override
1603     public int delete(String userid) {
1604         return this.memberDao.delete(userid);
1605     }
1606 }
1607
1608 3)MemberDaoImpl.java
1609
1610 package com.example.dao;
1611
1612 import java.sql.ResultSet;
1613 import java.sql.SQLException;
1614 import java.util.List;
1615
1616 import org.springframework.beans.factory.annotation.Autowired;
1617 import org.springframework.jdbc.core.JdbcTemplate;
1618 import org.springframework.jdbc.core.RowMapper;
1619 import org.springframework.stereotype.Repository;
1620
1621 import com.example.vo.MemberVO;
1622
1623 @Repository("memberDao")
1624 public class MemberDaoImpl implements MemberDao {
1625     @Autowired
1626     JdbcTemplate jdbcTemplate;
1627
1628     @Override
1629     public int create(MemberVO memberVo) {
1630         String sql = "INSERT INTO Member VALUES(?,?,?,?)";
1631         return this.jdbcTemplate.update(sql, memberVo.getUserid(), memberVo.getUsername(),
            memberVo.getAge(),
1632             memberVo.getGender(), memberVo.getCity());
1633     }
1634
1635     class MyRowMapper implements RowMapper<MemberVO> {
1636         @Override
1637         public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1638             MemberVO memberVo = new MemberVO(rs.getString("userid"), rs.getString("username"),
                rs.getInt("userage"),
1639                 rs.getString("gender"), rs.getString("city"));
1640             return memberVo;
1641         }
1642     }
1643
1644     @Override
1645     public MemberVO read(String userid) {
1646         String sql = "SELECT * FROM Member WHERE userid = ?";
1647         return this.jdbcTemplate.queryForObject(sql, new Object[] { userid }, new MyRowMapper());
1648     }
```



```

1649
1650     @Override
1651     public List<MemberVO> readAll() {
1652         String sql = "SELECT * FROM Member ORDER BY userid DESC";
1653         return this.jdbcTemplate.query(sql, new MyRowMapper());
1654     }
1655
1656     @Override
1657     public int update(MemberVO memberVo) {
1658         String sql = "UPDATE Member SET userage = ?, gender = ?, city = ? " + "WHERE userid = ?";
1659         return this.jdbcTemplate.update(sql, memberVo.getAge(), memberVo.getGender(),
            memberVo.getCity(),
            memberVo.getUserid());
1660     }
1661
1662
1663     @Override
1664     public int delete(String userid) {
1665         String sql = "DELETE FROM Member WHERE userid = ?";
1666         return this.jdbcTemplate.update(sql, userid);
1667     }
1668 }
1669
1670

```

```

1671 -----

```

1672 Task7. Form Data Validation

1673 1. Package Explorer > right-click > New > Other > Spring > Spring Legacy Project

1674 2. Select Spring MVC Project

1675 3. Project name : FormValidationDemo > Next

1676 4. Enter a topLevelPackage : com.example.biz > Finish

1677 5. pom.xml 수정하기

```

1678 <properties>
1679     <java-version>1.8</java-version>
1680     <org.springframework-version>5.2.0.RELEASE</org.springframework-version>
1681     <org.aspectj-version>1.9.4</org.aspectj-version>
1682     <org.slf4j-version>1.7.28</org.slf4j-version>
1683 </properties>
1684 ...
1685 <dependency>
1686     <groupId>javax.servlet</groupId>
1687     <artifactId>javax.servlet-api</artifactId>
1688     <version>4.0.1</version>
1689     <scope>provided</scope>
1690 </dependency>
1691 <dependency>
1692     <groupId>javax.servlet.jsp</groupId>
1693     <artifactId>javax.servlet.jsp-api</artifactId>
1694     <version>2.3.3</version>
1695     <scope>provided</scope>
1696 </dependency>
1697 <dependency>
1698     <groupId>junit</groupId>
1699     <artifactId>junit</artifactId>

```

```
1700     <version>4.12</version>
1701     <scope>test</scope>
1702 </dependency>
1703
1704 6. pom.xml > right-click > Run As > Maven install
1705 [INFO] BUILD SUCCESS
1706
1707 7. FormValidationDemo Project > right-click > Properties > Project Facets > Select Java > Change
    Version 1.8
1708     Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close
1709
1710 8. UserVO 객체 생성
1711     1)src/main/java/com.example.vo package 생성
1712     2)src/main/java/com.example.vo.UserVO class
1713
1714     package com.example.vo;
1715
1716     public class UserVO {
1717         private String name;
1718         private int age;
1719         private String userid;
1720         public String getName() {
1721             return name;
1722         }
1723         public void setName(String name) {
1724             this.name = name;
1725         }
1726         public int getAge() {
1727             return age;
1728         }
1729         public void setAge(int age) {
1730             this.age = age;
1731         }
1732         public String getUserid() {
1733             return userid;
1734         }
1735         public void setUserid(String userid) {
1736             this.userid = userid;
1737         }
1738         @Override
1739         public String toString() {
1740             return "UserVO [name=" + name + ", age=" + age + ", userid=" + userid + "]";
1741         }
1742     }
1743
1744
1745 9. Validator를 이용한 검증
1746     1)Data Command 객체에서 유효성 검사를 할 수 있다.
1747     2)UserValidator 객체 생성
1748     3)src/main/java/com.example.biz.UserValidator class
1749
1750     package com.example.biz;
```

```
1751
1752 import org.springframework.validation.Errors;
1753 import org.springframework.validation.Validator;
1754
1755 import com.example.vo.UserVO;
1756
1757 public class UserValidator implements Validator {
1758
1759     @Override
1760     public boolean supports(Class<?> arg0) {
1761         //검증할 객체의 class 타입 정보를 반환
1762         return UserVO.class.isAssignableFrom(arg0);
1763     }
1764
1765     @Override
1766     public void validate(Object obj, Errors errors) {
1767         System.out.println("검증시작");
1768         UserVO userVO = (UserVO)obj;
1769
1770         String username = userVO.getName();
1771         if(username == null || username.trim().isEmpty()) {
1772             System.out.println("이름의 값이 빠졌습니다.");
1773             errors.rejectValue("name", "No Value");
1774         }
1775
1776         int userage = userVO.getAge();
1777         if(userage == 0) {
1778             System.out.println("나이의 값이 빠졌습니다.");
1779             errors.rejectValue("age", "No Value");
1780         }
1781
1782         String userid = userVO.getUserid();
1783         if(userid == null || userid.trim().isEmpty()) {
1784             System.out.println("아이디의 값이 빠졌습니다.");
1785             errors.rejectValue("userid", "No Value");
1786         }
1787     }
1788 }
1789
1790 4)src/main/java/com.example.biz/HomeController.java
1791
1792 @RequestMapping(value = "/register", method=RequestMethod.GET)
1793 public String register() {
1794     return "register";
1795 }
1796
1797 @RequestMapping(value = "/register", method=RequestMethod.POST)
1798 public String register(@ModelAttribute("userVO") UserVO userVO, BindingResult result) {
1799     String page = "register_ok";
1800     UserValidator validator = new UserValidator();
1801     validator.validate(userVO, result);
1802     if(result.hasErrors()) {
```

```

1803         page = "register";
1804     }
1805     return page;
1806 }
1807
1808 5)src/main/webapp/WEB-INF/views/register.jsp
1809 <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
1810 <!DOCTYPE html>
1811 <html>
1812 <head>
1813 <meta charset="UTF-8">
1814 <title>회원 가입 폼</title>
1815 </head>
1816 <body>
1817     <form action="/biz/register" method="post">
1818         Name : <input type="text" name="name" /> <br />
1819         Age : <input type="number" name="age" /> <br />
1820         ID : <input type="text" name="userid" /> <br />
1821         <input type="submit" value="가입하기" />
1822     </form>
1823 </body>
1824 </html>
1825
1826 6)src/main/webapp/WEB-INF/views/register_ok.jsp
1827 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
1828 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
1829 <c:set var="user" value="${userVO}" />
1830 <!DOCTYPE html">
1831 <html>
1832 <head>
1833 <meta charset="UTF-8">
1834 <title>회원 가입 결과 창</title>
1835 </head>
1836 <body>
1837     <ul>
1838         <li>이름 : ${user.name}</li>
1839         <li>나이 : ${user.age}</li>
1840         <li>아이디 : ${user.userid}</li>
1841     </ul>
1842 </body>
1843 </html>
1844
1845 7)Test
1846 http://localhost:8080/biz/register에서
1847 이름, 나이, 아이디를 모두 입력하면 결과창으로 넘어오고
1848 한 개라도 입력하지 않으면 다시 입력창으로 간다.
1849
1850
1851 10. ValidationUtils class를 이용한 검증
1852 1)ValidationUtils class는 validate() method를 좀 더 편리하게 사용할 수 있게 해줌.
1853 2)UserValidator.java 수정
1854

```

```

1855     /*String username = userVO.getName();
1856     if(username == null || username.trim().isEmpty()) {
1857         System.out.println("이름의 값이 빠졌습니다.");
1858         errors.rejectValue("name", "No Value");
1859     }*/
1860
1861     ValidationUtils.rejectIfEmptyOrWhitespace(errors, "name", "No Value");
1862
1863 11. @Valid와 @InitBinder 이용하기
1864     1)Spring Framework이 대신 검증해 줌
1865     2)mvnrepository에서 'hibernate validator'로 검색
1866
1867     <dependency>
1868         <groupId>org.hibernate.validator</groupId>
1869         <artifactId>hibernate-validator</artifactId>
1870         <version>6.0.18.Final</version>
1871     </dependency>
1872
1873 3)pom.xml에 넣고 Maven Clean > Maven Install
1874 4)HomeController.java 수정
1875
1876     @RequestMapping(value = "/register", method=RequestMethod.POST)
1877     public String register(@ModelAttribute("userVO") @Valid UserVO userVO, BindingResult result) {
1878         String page = "register_ok";
1879         //UserValidator validator = new UserValidator();
1880         //validator.validate(userVO, result);
1881         if(result.hasErrors()) {
1882             page = "register";
1883         }
1884
1885         return page;
1886     }
1887
1888     @InitBinder
1889     protected void initBinder(WebDataBinder binder) {
1890         binder.setValidator(new UserValidator());
1891     }
1892
1893 12. Test
1894     http://localhost:8080/biz/register에서
1895     이름, 나이, 아이디를 모두 입력하면 결과창으로 넘어오고
1896     한 개라도 입력하지 않으면 다시 입력창으로 간다.
1897
1898
1899 -----
1900 Task8. Convert J2EE to Spring MVC
1901 1. In J2EE Perspective
1902 2. Project Explorer > right-click > New > Dynamic Web Project
1903 3. Project name : SpringWebDemo > Next > Check [Generate web.xml deployment descriptor] > Finish
1904 4. Convert to Maven Project
1905     1)project right-click > Configure > Convert to Maven Project > Finish
1906     2)Project : /SpringWebDemo

```

1907 3)Group Id : SpringWebDemo
1908 4)Artifact Id : SpringWebDemo
1909 5)version : 0.0.1-SNAPSHOT
1910 6)Packaging : war
1911 7)Finish
1912
1913 5. Add Spring Project Nature
1914 -project right-click > Spring Tools > Add Spring Project Nature
1915
1916 6. 새로 생성된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install
1917 <dependencies>
1918 <dependency>
1919 <groupId>org.springframework</groupId>
1920 <artifactId>spring-context</artifactId>
1921 <version>5.2.0.RELEASE</version>
1922 </dependency>
1923 <dependency>
1924 <groupId>junit</groupId>
1925 <artifactId>junit</artifactId>
1926 <version>4.12</version>
1927 <scope>test</scope>
1928 </dependency>
1929 <dependency>
1930 <groupId>org.springframework</groupId>
1931 <artifactId>spring-jdbc</artifactId>
1932 <version>5.2.0.RELEASE</version>
1933 </dependency>
1934 </dependencies>
1935
1936 7. Spring mvc library 검색 및 설치
1937 1)<http://mvnrepository.com>에서 'spring mvc'로 검색
1938 2)pom.xml에 추가
1939
1940 <dependency>
1941 <groupId>org.springframework</groupId>
1942 <artifactId>spring-webmvc</artifactId>
1943 <version>5.2.0.RELEASE</version>
1944 </dependency>
1945
1946 3)Maven Clean > Maven Install
1947
1948 8. Build path에 config folder 추가
1949 1)project right-click > Build Path > Configure Build Path > Select [Source] tab
1950 2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
1951 3)Folder name : config > Finish > OK > Apply and Close
1952 4)Java Resources > config 폴더 확인
1953
1954 9. config folder에 beans.xml file 생성
1955 1)Spring Perspective로 전환
1956 2)config > right-click > New > Other > Spring > Spring Bean Configuration File > beans.xml
1957 3)생성시 beans,context, mvc 체크
1958 <?xml version="1.0" encoding="UTF-8"?>

```

1959 <beans xmlns="http://www.springframework.org/schema/beans"
1960      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1961      xmlns:context="http://www.springframework.org/schema/context"
1962      xsi:schemaLocation="http://www.springframework.org/schema/beans
1963      http://www.springframework.org/schema/beans/spring-beans.xsd
1964      http://www.springframework.org/schema/context
1965      http://www.springframework.org/schema/context/spring-context-3.2.xsd">
1966
1967 </beans>
1968
1969 10. ContextLoaderListener class 설정
1970 1)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener -
1971 ContextLoaderListener] 를 선택하면 아래의 code가 자동 삽입
1972
1973 <!-- needed for ContextLoaderListener -->
1974 <context-param>
1975     <param-name>contextConfigLocation</param-name>
1976     <param-value>location</param-value>
1977 </context-param>
1978
1979 <!-- Bootstraps the root web application context before servlet initialization -->
1980 <listener>
1981     <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
1982 </listener>
1983
1984 2)아래 code로 변환
1985 <context-param>
1986     <param-name>contextConfigLocation</param-name>
1987     <param-value>classpath:beans.xml</param-value>
1988 </context-param>
1989
1990 11. DispatcherServlet Class 추가
1991 1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherservlet - DispatcherServlet
1992 declaration] 선택하면 아래의 code가 자동 추가된다.
1993
1994 <!-- The front controller of this Spring Web application, responsible for handling all application
1995 requests -->
1996 <servlet>
1997     <servlet-name>springDispatcherServlet</servlet-name>
1998     <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
1999     <init-param>
2000         <param-name>contextConfigLocation</param-name>
2001         <param-value>location</param-value>
2002     </init-param>
2003     <load-on-startup>1</load-on-startup>
2004 </servlet>
2005
2006 <!-- Map all requests to the DispatcherServlet for handling -->
2007 <servlet-mapping>
2008     <servlet-name>springDispatcherServlet</servlet-name>
2009     <url-pattern>url</url-pattern>
2010 </servlet-mapping>

```

```
2006
2007 2)아래의 code로 변환
2008     <init-param>
2009         <param-name>contextConfigLocation</param-name>
2010         <param-value>classpath:beans*.xml</param-value>
2011     </init-param>
2012
2013     <servlet-mapping>
2014         <servlet-name>springDispatcherServlet</servlet-name>
2015         <url-pattern>*.do</url-pattern>
2016     </servlet-mapping>
2017
2018 12. mvnrepository에서 'jstl'로 검색 후 설치
2019 1)목록에서 2번째 : 1.2버전
2020
2021     <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
2022     <dependency>
2023         <groupId>javax.servlet</groupId>
2024         <artifactId>jstl</artifactId>
2025         <version>1.2</version>
2026     </dependency>
2027
2028 2)pom.xml에 붙여넣고 Maven Clean > Maven Install
2029
2030
2031 13. Hello Controller 작성
2032 1)src/com.example.vo package 생성
2033 2)src/com.example.vo.HelloVO class 생성
2034
2035     package com.example.vo;
2036
2037     public class HelloVO {
2038         private String name;
2039
2040         public void setName(String name) {
2041             this.name = name;
2042         }
2043
2044         public String sayHello() {
2045             return "Hello " + name;
2046         }
2047     }
2048
2049 3)src/com.example.controller package 생성
2050 4)com.example.controller.HelloController class 생성
2051
2052     package com.example.controller;
2053
2054     import org.springframework.beans.factory.annotation.Autowired;
2055     import org.springframework.stereotype.Controller;
2056     import org.springframework.ui.Model;
2057     import org.springframework.web.bind.annotation.RequestMapping;
```



```

2058
2059     import com.example.vo.HelloVO;
2060
2061     @Controller
2062     public class HelloController {
2063         @Autowired
2064         private HelloVO helloBean;
2065
2066         @RequestMapping("/hello.do")
2067         public String hello(Model model) {
2068             String msg = helloBean.sayHello();
2069             model.addAttribute("greet", msg);
2070             return "hello.jsp";
2071         }
2072     }
2073
2074
2075 14. beans.xml 수정
2076     <context:component-scan base-package="com.example" />
2077
2078     <bean id="helloVO" class="com.example.vo.HelloVO">
2079         <property name="name" value="한지민" />
2080     </bean>
2081
2082 15. WebContent/hello.jsp 생성
2083
2084     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2085     <!DOCTYPE html>
2086     <html>
2087     <head>
2088         <meta charset="UTF-8">
2089         <title>Insert title here</title>
2090     </head>
2091     <body>
2092         ${greet}
2093     </body>
2094 </html>
2095
2096 16. project > right-click > Run As > Run on Server > Finish
2097
2098 17. http://localhost:8080/SpringWebDemo/hello.do
2099     Hello 한지민
2100
2101
2102 -----
2103 Task9. Convert J2EE to Spring MVC
2104 1. In J2EE Perspective
2105 2. Project Explorer > right-click > New > Dynamic Web Project
2106 3. Project name : SpringWebDemo1 > Next > Check [Generate web.xml deployment descriptor] > Finish
2107 4. Convert to Maven Project
2108     -project right-click > Configure > Convert to Maven Project > Finish
2109 5. Add Spring Project Nature

```

2110 -project right-click > Spring Tools > Add Spring Project Nature
2111
2112 6. 새로 생성된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install
2113 <dependencies>
2114 <dependency>
2115 <groupId>org.springframework</groupId>
2116 <artifactId>spring-context</artifactId>
2117 <version>5.2.0.RELEASE</version>
2118 </dependency>
2119 <dependency>
2120 <groupId>junit</groupId>
2121 <artifactId>junit</artifactId>
2122 <version>4.12</version>
2123 <scope>test</scope>
2124 </dependency>
2125 <dependency>
2126 <groupId>org.springframework</groupId>
2127 <artifactId>spring-jdbc</artifactId>
2128 <version>5.2.0.RELEASE</version>
2129 </dependency>
2130 <dependency>
2131 <groupId>javax.servlet</groupId>
2132 <artifactId>jstl</artifactId>
2133 <version>1.2</version>
2134 </dependency>
2135 <dependency>
2136 <groupId>com.oracle</groupId>
2137 <artifactId>ojdbc6</artifactId>
2138 <version>11.2</version>
2139 </dependency>
2140 <dependency>
2141 <groupId>org.springframework</groupId>
2142 <artifactId>spring-webmvc</artifactId>
2143 <version>5.2.0.RELEASE</version>
2144 </dependency>
2145 </dependencies>
2146
2147 2)Maven Clean > Maven Install
2148
2149
2150 7. Build path에 config folder 추가
2151 1)project right-click > Build Path > Configure Build Path > Select [Source] tab
2152 2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
2153 3)Folder name : config > Finish > OK > Apply and Close
2154 4)Java Resources > config 폴더 확인
2155
2156
2157 8. config folder에 beans.xml file 생성
2158 1)Spring Perspective로 전환
2159 2)config Folder > right-click > New > Spring Bean Configuration File
2160 3)File name : beans.xml
2161 4)생성시 beans,context, mvc 체크

```

2162 <?xml version="1.0" encoding="UTF-8"?>
2163 <beans xmlns="http://www.springframework.org/schema/beans"
2164       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2165       xmlns:context="http://www.springframework.org/schema/context"
2166       xsi:schemaLocation="http://www.springframework.org/schema/beans
2167       http://www.springframework.org/schema/beans/spring-beans.xsd
2168       http://www.springframework.org/schema/context
2169       http://www.springframework.org/schema/context/spring-context-3.2.xsd">
2170
2171 </beans>
2172

```

2173 9. ContextLoaderListener class 설정

2174 1)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener - ContextLoaderListener] 를 선택하면 아래의 코드가 자동 삽입

```

2175
2176 <!-- needed for ContextLoaderListener -->
2177 <context-param>
2178   <param-name>contextConfigLocation</param-name>
2179   <param-value>location</param-value>
2180 </context-param>
2181
2182 <!-- Bootstraps the root web application context before servlet initialization -->
2183 <listener>
2184   <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
2185 </listener>
2186

```

2187 2)아래 코드로 변환

```

2188 <context-param>
2189   <param-name>contextConfigLocation</param-name>
2190   <param-value>classpath:beans.xml</param-value>
2191 </context-param>
2192
2193

```

2194 10. DispatcherServlet Class 추가

2195 1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherservlet - DispatcherServlet declaration] 선택하면 아래의 코드가 자동 추가된다.

```

2196
2197 <!-- The front controller of this Spring Web application, responsible for handling all application
2198 requests -->
2199 <servlet>
2200   <servlet-name>springDispatcherServlet</servlet-name>
2201   <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
2202   <init-param>
2203     <param-name>contextConfigLocation</param-name>
2204     <param-value>location</param-value>
2205   </init-param>
2206   <load-on-startup>1</load-on-startup>
2207 </servlet>
2208
2209 <!-- Map all requests to the DispatcherServlet for handling -->

```

```
2209     <servlet-mapping>
2210         <servlet-name>springDispatcherServlet</servlet-name>
2211         <url-pattern>url</url-pattern>
2212     </servlet-mapping>
2213
2214 2)아래의 코드로 변환
2215     <init-param>
2216         <param-name>contextConfigLocation</param-name>
2217         <param-value>classpath:beans*.xml</param-value>
2218     </init-param>
2219
2220     <servlet-mapping>
2221         <servlet-name>springDispatcherServlet</servlet-name>
2222         <url-pattern>*.do</url-pattern>
2223     </servlet-mapping>
2224
2225
2226 11. UserVO class 생성
2227 1)src/com.example.vo package 생성
2228 2)src/com.example.vo.UserVO class 생성
2229
2230     package com.example.vo;
2231
2232     public class UserVO {
2233         private String userId;
2234         private String name;
2235         private String gender;
2236         private String city;
2237         public UserVO() {}
2238         public UserVO(String userId, String name, String gender, String city) {
2239             this.userId = userId;
2240             this.name = name;
2241             this.gender = gender;
2242             this.city = city;
2243         }
2244         public String getUserId() {
2245             return userId;
2246         }
2247         public void setUserId(String userId) {
2248             this.userId = userId;
2249         }
2250         public String getName() {
2251             return name;
2252         }
2253         public void setName(String name) {
2254             this.name = name;
2255         }
2256         public String getGender() {
2257             return gender;
2258         }
2259         public void setGender(String gender) {
2260             this.gender = gender;
```

```

2261     }
2262     public String getCity() {
2263         return city;
2264     }
2265     public void setCity(String city) {
2266         this.city = city;
2267     }
2268     @Override
2269     public String toString() {
2270         return "UserVO [userId=" + userId + ", name=" + name + ", gender=" + gender + ", city=" +
2271             city + "]";
2272     }
2273
2274
2275 12. UserDao 객체 생성
2276 1)src/com.example.dao package 생성
2277 2)src/com.example.dao.UserDao interface
2278
2279     package com.example.dao;
2280
2281     import java.util.List;
2282
2283     import com.example.vo.UserVO;
2284
2285     public interface UserDao {
2286         void insert(UserVO user);
2287
2288         List<UserVO> readAll();
2289
2290         void update(UserVO user);
2291
2292         void delete(String id);
2293
2294         UserVO read(String id);
2295     }
2296
2297 -src/com.example.dao.UserDaoImplJDBC.java 생성
2298
2299     package com.example.dao;
2300
2301     import java.sql.ResultSet;
2302     import java.sql.SQLException;
2303     import java.util.List;
2304
2305     import javax.sql.DataSource;
2306
2307     import org.springframework.beans.factory.annotation.Autowired;
2308     import org.springframework.dao.EmptyResultDataAccessException;
2309     import org.springframework.jdbc.core.JdbcTemplate;
2310     import org.springframework.jdbc.core.RowMapper;
2311     import org.springframework.stereotype.Repository;

```

```
2312
2313 import com.example.vo.UserVO;
2314
2315 @Repository("userDao")
2316 public class UserDaoImplJDBC implements UserDao {
2317
2318     private JdbcTemplate jdbcTemplate;
2319
2320     @Autowired
2321     public void setDataSource(DataSource dataSource) {
2322         this.jdbcTemplate = new JdbcTemplate(dataSource);
2323     }
2324
2325     class UserMapper implements RowMapper<UserVO> {
2326         public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
2327             UserVO user = new UserVO();
2328             user.setUserId(rs.getString("userId"));
2329             user.setName(rs.getString("name"));
2330             user.setGender(rs.getString("gender"));
2331             user.setCity(rs.getString("city"));
2332             return user;
2333         }
2334     }
2335
2336     @Override
2337     public void insert(UserVO user) {
2338         String SQL = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
2339         jdbcTemplate.update(SQL, user.getUserId(), user.getName(), user.getGender(), user.getCity());
2340
2341         System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" + user.getName());
2342     }
2343
2344     @Override
2345     public List<UserVO> readAll() {
2346         String SQL = "SELECT * FROM users";
2347         List<UserVO> userList = jdbcTemplate.query(SQL, new UserMapper());
2348         return userList;
2349     }
2350
2351     @Override
2352     public void update(UserVO user) {
2353         String SQL = "UPDATE users SET name = ?, gender = ?, city = ? WHERE userid = ?";
2354         jdbcTemplate.update(SQL, user.getName(), user.getGender(), user.getCity(), user.getUserId());
2355         System.out.println("갱신된 Record with ID = " + user.getUserId());
2356     }
2357
2358     @Override
2359     public void delete(String id) {
2360         String SQL = "DELETE FROM users WHERE userid = ?";
2361         jdbcTemplate.update(SQL, id);
2362         System.out.println("삭제된 Record with ID = " + id);
2363     }
```

```

2364
2365     @Override
2366     public UserVO read(String id) {
2367         String SQL = "SELECT * FROM users WHERE userid = ?";
2368         try {
2369             UserVO user = jdbcTemplate.queryForObject(SQL, new Object[] { id }, new UserMapper());
2370             return user;
2371         } catch (EmptyResultDataAccessException e) {
2372             return null;
2373         }
2374     }
2375 }

```

2376

2377

2378 13. UserService 객체 생성

2379 1)src/com.example.service package 생성

2380 2)src/com.example.service.UserService interface

2381

2382 package com.example.service;

2383

2384 import java.util.List;

2385

2386 import com.example.vo.UserVO;

2387

2388 public interface UserService {

2389 void insertUser(UserVO user);

2390

2391 List<UserVO> getUserList();

2392

2393 void deleteUser(String id);

2394

2395 UserVO getUser(String id);

2396

2397 void updateUser(UserVO user);

2398 }

2399

2400 3)src/com.example.service.UserServiceImpl.java

2401

2402 package com.example.service;

2403

2404 import java.util.List;

2405

2406 import org.springframework.beans.factory.annotation.Autowired;

2407 import org.springframework.stereotype.Service;

2408

2409 import com.example.dao.UserDao;

2410 import com.example.vo.UserVO;

2411

2412 @Service("userService")

2413 public class UserServiceImpl implements UserService {

2414 @Autowired

2415 UserDao userDao;

```
2416
2417     @Override
2418     public void insertUser(UserVO user) {
2419         this.userDao.insert(user);
2420     }
2421
2422     @Override
2423     public List<UserVO> getUserList() {
2424         return this.userDao.readAll();
2425     }
2426
2427     @Override
2428     public void deleteUser(String id) {
2429         this.userDao.delete(id);
2430     }
2431
2432     @Override
2433     public UserVO getUser(String id) {
2434         return this.userDao.read(id);
2435     }
2436
2437     @Override
2438     public void updateUser(UserVO user) {
2439         this.userDao.update(user);
2440     }
2441 }
2442
2443
2444 14. UserController 객체 생성
2445     1)src/com.example.controller package 생성
2446     2)com.example.controller.UserController class 생성
2447
2448     package com.example.controller;
2449
2450     import org.springframework.beans.factory.annotation.Autowired;
2451     import org.springframework.stereotype.Controller;
2452
2453     import com.example.service.UserService;
2454
2455     @Controller
2456     public class UserController {
2457         @Autowired
2458         private UserService userService;
2459
2460         @RequestMapping("/userInfo.do")
2461         public String getUserList(@RequestParam("userId") String userId, Model model) {
2462             UserVO user = userService.getUser(userId);
2463             //System.out.println(user);
2464             model.addAttribute("user", user);
2465             return "userInfo.jsp";
2466         }
2467     }
```



```
2468
2469
2470 15. config/dbinfo.properties file 생성
2471
2472 db.driverClass=oracle.jdbc.driver.OracleDriver
2473 db.url=jdbc:oracle:thin:@localhost:1521:XE
2474 db.username=hr
2475 db.password=hr
2476
2477
2478 16. beans.xml 수정
2479
2480 <context:component-scan base-package="com.example" />
2481
2482 <context:property-placeholder location="classpath:dbinfo.properties" />
2483 <bean id="dataSource" class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
2484     <property name="driverClass" value="${db.driverClass}" />
2485     <property name="url" value="${db.url}" />
2486     <property name="username" value="${db.username}" />
2487     <property name="password" value="${db.password}" />
2488 </bean>
2489
2490
2491 17. WebContent/index.jsp 생성
2492
2493 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2494 <c:redirect url="userInfo.do" />
2495
2496
2497 18. WebContent/userInfo.jsp 생성
2498
2499 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2500 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2501 <c:set var="user" value="${user}" />
2502 <!DOCTYPE html>
2503 <html>
2504 <head>
2505 <meta charset="UTF-8">
2506 <title>Insert title here</title>
2507 </head>
2508 <body>
2509     <h1>userInfo.jsp</h1>
2510     <h2>사용자 정보</h2>
2511     아이디 : ${user.userId} <br />
2512     이름 : ${user.name} <br />
2513     성별 : ${user.gender} <br />
2514     도시 : ${user.city} <br />
2515 </body>
2516 </html>
2517
2518
2519 19. project > right-click > Run As > Run on Server > Finish
```

2520
2521 20. <http://localhost:8080/SpringWebDemo/userinfo.do?userId=scott>
2522
2523
2524 -----
2525 Task10. File Upload with Spring MVC
2526 1. In J2EE Perspective
2527 2. Project Explorer > right-click > New > Dynamic Web Project
2528 3. Project name : FileUploadDemo > Next > Check [Generate web.xml deployment descriptor] > Finish
2529 4. Convert to Maven Project
2530 1)project right-click > Configure > Convert to Maven Project > Finish
2531
2532 5. Add Spring Project Nature
2533 1)project right-click > Spring Tools > Add Spring Project Nature
2534
2535 6. 새로 생성된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install
2536 <dependencies>
2537 <dependency>
2538 <groupId>org.springframework</groupId>
2539 <artifactId>spring-context</artifactId>
2540 <version>5.2.0.RELEASE</version>
2541 </dependency>
2542 <dependency>
2543 <groupId>org.springframework</groupId>
2544 <artifactId>spring-webmvc</artifactId>
2545 <version>5.2.0.RELEASE</version>
2546 </dependency>
2547 </dependencies>
2548
2549
2550 7. ContextLoaderListener class 설정
2551 1)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener -
ContextLoaderListener] 를 선택하면 아래의 코드가 자동 삽입
2552
2553 <!-- needed for ContextLoaderListener -->
2554 <context-param>
2555 <param-name>contextConfigLocation</param-name>
2556 <param-value>location</param-value>
2557 </context-param>
2558
2559 <!-- Bootstraps the root web application context before servlet initialization -->
2560 <listener>
2561 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
2562 </listener>
2563
2564 2)아래 코드로 변환
2565 <context-param>
2566 <param-name>contextConfigLocation</param-name>
2567 <param-value>classpath:applicationContext.xml</param-value>
2568 </context-param>
2569
2570

2571 8. DispatcherServlet Class 추가

2572 1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherServlet - DispatcherServlet
2573 declaration] 선택하면 아래의 코드가 자동 추가된다.

```
2574 <!-- The front controller of this Spring Web application, responsible for handling all application  
requests -->
```

```
2575 <servlet>
```

```
2576 <servlet-name>springDispatcherServlet</servlet-name>
```

```
2577 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
```

```
2578 <init-param>
```

```
2579 <param-name>contextConfigLocation</param-name>
```

```
2580 <param-value>location</param-value>
```

```
2581 </init-param>
```

```
2582 <load-on-startup>1</load-on-startup>
```

```
2583 </servlet>
```

```
2584
```

```
2585 <!-- Map all requests to the DispatcherServlet for handling -->
```

```
2586 <servlet-mapping>
```

```
2587 <servlet-name>springDispatcherServlet</servlet-name>
```

```
2588 <url-pattern>url</url-pattern>
```

```
2589 </servlet-mapping>
```

```
2590
```

2591 2)아래의 코드로 변환

```
2592 <init-param>
```

```
2593 <param-name>contextConfigLocation</param-name>
```

```
2594 <param-value>classpath:beans.xml</param-value>
```

```
2595 </init-param>
```

```
2596
```

```
2597 <servlet-mapping>
```

```
2598 <servlet-name>springDispatcherServlet</servlet-name>
```

```
2599 <url-pattern>/</url-pattern>
```

```
2600 </servlet-mapping>
```

```
2601
```

```
2602
```

2603 9. FileUpload library 추가

2604 1)Apache에서 제공하는 Common FileUpload library를 사용하여 file upload를 처리하기 위한 library

2605 2)mvnrepository에서 'common fileupload'라고 검색하여 library 추가

```
2606 <dependency>
```

```
2607 <groupId>commons-fileupload</groupId>
```

```
2608 <artifactId>commons-fileupload</artifactId>
```

```
2609 <version>1.4</version>
```

```
2610 </dependency>
```

```
2611
```

2612 3)mvnrepository에서 'commons io'라고 검색하여 library 추가

```
2613 <dependency>
```

```
2614 <groupId>commons-io</groupId>
```

```
2615 <artifactId>commons-io</artifactId>
```

```
2616 <version>2.6</version>
```

```
2617 </dependency>
```

```
2618
```

2619 4)Maven Clean > Maven Install

```
2620
```

2621
2622 10. Thumbnail Image Library 추가
2623 1)mvnrepository에서 'imgscalr-lib'라고 검색하여 library 추가
2624 <dependency>
2625 <groupId>org.imgscalr</groupId>
2626 <artifactId>imgscalr-lib</artifactId>
2627 <version>4.2</version>
2628 </dependency>
2629
2630 2)Maven Clean > Maven Install
2631
2632
2633 11. Build path에 config Foler 추가
2634 1)project right-click > Build Path > Configure Build Path > Select [Source] tab
2635 2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
2636 3)Folder name : config > Finish > OK > Apply and Close
2637 4)Java Resources > config 폴더 확인
2638
2639
2640 12. config Folder에 applicationContext.xml file 생성
2641 1)config > right-click > New > Other > Spring > Spring Bean Configuration File
2642 2)Name : applicationContext.xml
2643 3)Namespace Tab에서 context, mvc check 할 것
2644
2645 <?xml version="1.0" encoding="UTF-8"?>
2646 <beans xmlns="<http://www.springframework.org/schema/beans>"
2647 xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"
2648 xmlns:context="<http://www.springframework.org/schema/context>"
2649 xmlns:mvc="<http://www.springframework.org/schema/mvc>"
2650 xsi:schemaLocation="<http://www.springframework.org/schema/mvc>
<http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd>
2651 <http://www.springframework.org/schema/beans>
<http://www.springframework.org/schema/beans/spring-beans.xsd>
2652 <http://www.springframework.org/schema/context>
<http://www.springframework.org/schema/context/spring-context-4.3.xsd>">
2653
2654 <context:component-scan
2655 base-package="com.example" />
2656 <mvc:annotation-driven />
2657 </beans>
2658
2659
2660 13. config Folder에 beans.xml file 생성
2661 1)config > right-click > New > Other > Spring > Spring Bean Configuration File
2662 2)Name : beans.xml
2663 3)beans.xml에 Spring multipartResolver 추가
2664
2665 <bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
2666 <property name="maxUploadSize" value="10240000" />
2667 <property name="defaultEncoding" value="utf-8" />
2668 </bean>

```
2669
2670
2671 14. web.xml에 한글 File Encoding 처리하기
2672 1)한글 File이 Upload될 때 File 명이 깨지는 것을 해결하기 위해 web.xml에 아래 내용을 추가한다.
2673
2674     <filter>
2675         <filter-name>encodingFilter</filter-name>
2676         <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
2677         <init-param>
2678             <param-name>encoding</param-name>
2679             <param-value>UTF-8</param-value>
2680         </init-param>
2681         <init-param>
2682             <param-name>forceEncoding</param-name>
2683             <param-value>true</param-value>
2684         </init-param>
2685     </filter>
2686     <filter-mapping>
2687         <filter-name>encodingFilter</filter-name>
2688         <url-pattern>/*</url-pattern>
2689     </filter-mapping>
2690
2691
2692 15. View 작성
2693 1)WebContent/form.jsp
2694
2695     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2696     <!DOCTYPE html>
2697     <html>
2698     <head>
2699     <meta charset="UTF-8">
2700     <title>Insert title here</title>
2701     </head>
2702     <body>
2703         <h1>file 업로드 예제</h1>
2704         <form method="post" action="upload" enctype="multipart/form-data">
2705             <label>email:</label> <input type="text" name="email"> <br>
2706             <br> <label>file:</label> <input type="file" name="file1">
2707             <br>
2708             <br> <input type="submit" value="upload">
2709         </form>
2710     </body>
2711 </html>
2712
2713
2714 16. Service 작성
2715 1)src/com.example.service package
2716 2)src/com.example.service.FileUploadService.java
2717
2718     package com.example.service;
2719
2720     import java.io.FileOutputStream;
```

```
2721 import java.io.IOException;
2722 import java.util.Calendar;
2723
2724 import org.springframework.stereotype.Service;
2725 import org.springframework.web.multipart.MultipartFile;
2726
2727 @Service
2728 public class FileUploadService {
2729     // 리눅스 기준으로 file 경로를 작성 ( 루트 경로인 /으로 시작한다. )
2730     // 윈도우라면 workspace의 드라이브를 파악하여 JVM이 알아서 처리해준다.
2731     // 따라서 workspace가 C드라이브에 있다면 C드라이브에 upload 폴더를 생성해 놓아야 한다.
2732     private static final String SAVE_PATH = "/upload";
2733     private static final String PREFIX_URL = "/upload/";
2734
2735     public String restore(MultipartFile multipartFile) {
2736         String uri = null;
2737
2738         try {
2739             // file 정보
2740             String originFilename = multipartFile.getOriginalFilename();
2741             String extName = originFilename.substring(originFilename.lastIndexOf("."),
2742             originFilename.length());
2743             Long size = multipartFile.getSize();
2744
2745             // 서버에서 저장 할 file 이름
2746             String saveFileName = genSaveFileName(extName);
2747
2748             System.out.println("originFilename : " + originFilename);
2749             System.out.println("extensionName : " + extName);
2750             System.out.println("size : " + size);
2751             System.out.println("saveFileName : " + saveFileName);
2752
2753             writeFile(multipartFile, saveFileName);
2754             uri = PREFIX_URL + saveFileName;
2755         } catch (IOException e) {
2756             // 원래라면 RuntimeException 을 상속받은 예외가 처리되어야 하지만
2757             // 편의상 RuntimeException을 던진다.
2758             // throw new FileUploadException();
2759             throw new RuntimeException(e);
2760         }
2761         return uri;
2762     }
2763
2764     // 현재 시간을 기준으로 file 이름 생성
2765     private String genSaveFileName(String extName) {
2766         String fileName = "";
2767
2768         Calendar calendar = Calendar.getInstance();
2769         fileName += calendar.get(Calendar.YEAR);
2770         fileName += calendar.get(Calendar.MONTH);
2771     }
```

```

2772     fileName += calendar.get(Calendar.DATE);
2773     fileName += calendar.get(Calendar.HOUR);
2774     fileName += calendar.get(Calendar.MINUTE);
2775     fileName += calendar.get(Calendar.SECOND);
2776     fileName += calendar.get(Calendar.MILLISECOND);
2777     fileName += extName;
2778
2779     return fileName;
2780 }
2781
2782
2783 // file을 실제로 write 하는 메서드
2784 private boolean writeFile(MultipartFile multipartFile, String saveFileName)
2785     throws IOException{
2786     boolean result = false;
2787
2788     byte[] data = multipartFile.getBytes();
2789     FileOutputStream fos = new FileOutputStream(SAVE_PATH + "/" + saveFileName);
2790     fos.write(data);
2791     fos.close();
2792
2793     return result;
2794 }
2795 }

```

- 2796
- 2797 3)SAVE_PATH는 file을 저장할 위치를 가리킨다.
- 2798 -일반적으로 server는 Linux 기반이므로 Linux 경로명을 사용하는 것이 좋다.
- 2799 -즉 file을 root 경로인 / 아래의 upload folder에 저장하겠다는 의미인데, Windows에서는 JVM이 알아서 workspace가 존재하는 drive의 위치를 찾아서 drive를 root 경로로 하여 upload folder에 저장한다.
- 2800 -예를들어 Eclipse workspace가 C drive에 있다면 C drive의 upload folder에 file이 저장될 것이다.
- 2801 4)PREFIX_URL은 저장된 file을 JSP에서 불러오기 위한 경로를 의미한다.
- 2802 5)MultipartFile 객체는 file의 정보를 담고 있다.
- 2803 6)uri을 반환하는 이유는 view page에서 바로 image file을 보기 위함이다.
- 2804 -만약 DB에서 image 경로를 저장 해야 한다면, 이와 같이 uri을 반환하면 좋을 것이다.
- 2805 7)현재 시간을 기준으로 file 이름을 바꾼다.
- 2806 -이렇게 하는 이유는, 여러 사용자가 올린 file의 이름이 같을 경우 덮어 씌어지는 문제가 발생하기 때문이다.
- 2807 -따라서 file 이름이 중복될 수 있는 문제를 해결하기 위해 ms단위의 시스템 시간을 이용하여 file 이름을 변경한다.
- 2808 8)FileOutputStream 객체를 이용하여 file을 저장한다.

2809

2810

2811 17. Controller 작성

```

2812 1)com.example.controller package
2813 2)com.example.controller.FileUploadController.java
2814
2815     package com.example.controller;
2816
2817     import org.springframework.beans.factory.annotation.Autowired;
2818     import org.springframework.stereotype.Controller;
2819     import org.springframework.ui.Model;
2820     import org.springframework.web.bind.annotation.RequestMapping;

```

```

2821 import org.springframework.web.bind.annotation.RequestMethod;
2822 import org.springframework.web.bind.annotation.RequestParam;
2823 import org.springframework.web.multipart.MultipartFile;
2824
2825 import com.example.service.FileUploadService;
2826
2827 @Controller
2828 public class FileUploadController {
2829     @Autowired
2830     FileUploadService fileUploadService;
2831
2832     @RequestMapping("/form")
2833     public String form() {
2834         return "form.jsp";
2835     }
2836
2837     @RequestMapping(value = "/upload", method = RequestMethod.POST)
2838     public String upload(@RequestParam("email") String email, @RequestParam("file1") MultipartFile
2839         file, Model model) {
2840         String uri = fileUploadService.restore(file);
2841         model.addAttribute("uri", uri);
2842         return "result.jsp";
2843     }
2844 }

```

2846 18. WebContent/result.jsp

```

2847 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2848 <!DOCTYPE html>
2849 <html>
2850 <head>
2851 <meta charset="UTF-8">
2852 <title>Insert title here</title>
2853 </head>
2854 <body>
2855 <h1>Upload completed</h1>
2856 <div class="result-images">
2857 
2858 </div>
2859 <p>
2860 <a href='/FileUploadDemo/form'> 다시 업로드 하기 </a>
2861 </p>
2862 </body>
2863 </html>

```

2866 19. C:/(현재 workspace가 C:라면)upload Folder 생성할 것

2868 20. Project > right-click > Run As > Run on Server

2869 <http://localhost:8080/FileUploadDemo/form>

2870
2871

2872 21. 문제점 및 해결

- 2873 1)Upload Folder(C:/upload)를 보면 File이 Upload된 것을 확인할 수 있지만, 결과 화면을 보면 Image가 제대로 출력 되지 않을 것이다.
- 2874 2)Image File을 right-click하여 경로를 보면 아마 다음과 같을 것이다.
- 2875 3)<http://localhost:8080/FileUploadDemo/upload/업로드한 파일>
- 2876 4)File을 저장할 때 [upload]라는 Folder에 저장을 했는데, File을 저장할 때의 Upload는 C Drive 내의 [upload] Folder이고,
- 2877 5)위 URL에서 [upload]는 Application 상 경로에 있는 upload이므로 WEB-INF 폴더의 하위 folder로서의 upload를 의미한다.
- 2878 6)즉 실제 File이 저장된 Server 상의 위치(물리 주소)와 Application에서 보여주고자 하는 File 경로(가상 주소)가 일치하지 않은 것이다.
- 2879 7)따라서 실제 File이 저장되어 있는 위치와 Application 상의 위치를 일치시키는 작업이 필요하다.
- 2880 8)beans.xml에 물리 주소와 가상 주소를 mapping 해주는 code를 추가하도록 해야한다.

```
2881
2882     <!-- resource mapping -->
2883     <!-- location : 물리적 주소 / mapping : 가상 주소 -->
2884     <mvc:resources location="file:///C:/upload/" mapping="/upload/*"/>
2885
```

- 2886 9)이제 정상적으로 result.jsp에서 image가 출력될 것이다.

2889 22. Multiple File Upload

- 2890 1)이번에는 여러 개의 File을 Upload 할 수 있는 Multiple Upload를 알아보자.
- 2891 2)수정할 부분은 <input> tag와 Controller에서 MultipartFile 객체를 받는 Parameter 부분 두 곳인데, 필요한 부분만 보자.

2892 3)form.jsp

```
2893
2894     <input type="file" name="files" multiple>
2895
```

- 2896 4)<input> 태그에서는 multiple 속성만 추가하면 된다.
- 2897 5)"File선택"을 클릭하면 ctrl 키를 눌러서 여러 개의 File을 선택할 수 있다.

2899 6)FileUploadController

```
2900
2901     @RequestMapping( "/upload" )
2902     public String upload(@RequestParam String email,
2903         @RequestParam(required=false) List<MultipartFile> files, Model model) {
2904
2905         ...
2906
2907     }
```

- 2909 7)Controller에서는 여러 개의 File을 받기 때문에 MultipartFile을 List로 받아야 한다.