

```

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
9  5. pom.xml 수정하기
10 <properties>
11     <java-version>11</java-version>
12     <org.springframework-version>5.3.12</org.springframework-version>
13     <org.aspectj-version>1.9.7</org.aspectj-version>
14     <org.slf4j-version>1.7.32</org.slf4j-version>
15 </properties>
16 ...
17 <dependency>
18     <groupId>javax.servlet</groupId>
19     <artifactId>javax.servlet-api</artifactId>
20     <version>4.0.1</version>
21     <scope>provided</scope>
22 </dependency>
23 <dependency>
24     <groupId>javax.servlet.jsp</groupId>
25     <artifactId>javax.servlet.jsp-api</artifactId>
26     <version>2.3.3</version>
27     <scope>provided</scope>
28 </dependency>
29 <dependency>
30     <groupId>org.junit.jupiter</groupId>
31     <artifactId>junit-jupiter-api</artifactId>
32     <version>5.8.1</version>
33     <scope>test</scope>
34 </dependency>
35 ...
36 <build>
37     <plugins>
38         <plugin>
39             <artifactId>maven-eclipse-plugin</artifactId>
40             <version>2.10</version>
41         ...
42     </plugin>
43     <plugin>
44         <groupId>org.apache.maven.plugins</groupId>
45         <artifactId>maven-compiler-plugin</artifactId>
46         <version>3.8.1</version>
47         <configuration>
48             <source>11</source>
49             <target>11</target>
50         ...
51     </plugin>
52     <plugin>
53         <groupId>org.codehaus.mojo</groupId>
54         <artifactId>exec-maven-plugin</artifactId>
55         <version>3.0.0</version>
56
57 6. pom.xml > right-click > Run As > Maven install
58 [INFO] BUILD SUCCESS
59
60
61 7. HelloWorldWeb project > right-click > Properties > Project Facets > Select Java > Change Version 11
62 -Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close
63
64
65
66 8. HelloWorldWeb Project right-click > Run As > Run on Server > Finish
67
68
69 9. http://localhost:8080/biz/
70
71 Hello world!
72
73 The time on the server is 2019년 6월 11일 (화) 오후 11시 40분 58초.<--원래 한글 깨짐
74

```

```

75
76 10. 한글 깨짐을 수정하는 것은 src/main/webapp/WEB-INF/views/home.jsp에서
77 <%@ page session="false" pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>로 수정
78
79
80 11. Context name 변경하기
81 1)Package Explorer에서 Servers/Tomcat v9.0 Server at localhost-config/server.xml에서 다음과 같이 수정한다.
82 -path="/biz" --> path="/demo"
83 <Context docBase="HelloWorldWeb" path="/demo" reloadable="true"
      source="org.eclipse.jst.jee.server:HelloWorldWeb"/>
84
85 2)수정 후 restart 하면 http://localhost:8080/biz --> http://localhost:8080/demo 로 변경됨
86
87
88 -----
89 Task2. resources Folder 이용하기
90 1. Image 경로 알아내기
91 1)src/main/webapp/resources/에 images Folder를 STS Package Explorer에서 생성한다.
92 2)Download받은 image를 src/main/webapp/resources/images/에 넣는다.
93 3)home.jsp에 아래 code를 추가한다.
94 <p></p>
95
96 4)Image가 잘 나온다.
97
98
99 2. Image 경로 변경
100 1)apple.jpg image 경로를 src/main/webapp/images/로 이동.
101 2)하지만 이렇게 하면 image가 보이지 않는다.
102 3)왜냐하면, servlet-context.xml에서 resource의 경로는 <resources mapping="/resources/**" location="/resources/"
    />이기 때문.
103 4)즉, 기본적으로 resources folder 아래에서 resource를 찾는다.
104
105
106 3. <resources />추가
107 1)다시 resources Folder 하위로 images Folder를 복사
108 2)/src/main/webapp/하위에 images Folder를 생성하고 image를 넣고 home.jsp에 아래의 code를 추가한다.
109 <p></p>
110 <p></p>
111
112 3)하지만 아래의 image는 보이지 않는다.
113 4)왜냐하면 새로 추가한 images Folder는 servlet-context.xml에서 설정하지 않았기 때문.
114 5)Image를 보이게 하기 위해 servlet-context.xml에 아래의 Code를 추가한다.
115 <resources mapping="/resources/**" location="/resources/" />
116 <resources mapping="/images/**" location="/images/" />
117
118 6)src/main/webapp/images Folder 추가
119 7)Project right-click > Run As > Run on Server > Restart >
120 -Image가 제대로 2개가 나온다.
121
122
123 -----
124 Task3. Controller Class 제작하기
125 1. 제작순서
126 1)@Controller를 이용한 class 생성
127 2)@RequestMapping을 이용한 요청 경로 지정
128 3)요청 처리 method 구현
129 4)View 이름 return
130
131 5)src/main/java/com.example.biz.UserController class 생성
132
133 @Controller
134 public class UserController {
135
136
137 2. 요청 처리 method 생성
138
139 package com.example.biz;
140
141 import org.springframework.stereotype.Controller;
142 import org.springframework.ui.Model;
143 import org.springframework.web.bind.annotation.RequestMapping;
144 import org.springframework.web.bind.annotation.RequestMethod;
145 import org.springframework.web.servlet.ModelAndView;
146

```

```

147 @Controller
148 public class UserController {
149     @RequestMapping("/view")
150     public String view(Model model){
151         /*
152         model.addAttribute("username", "한지민");
153         model.addAttribute("userage", 24);
154         model.addAttribute("job", "Developer");
155         return "view";
156         */
157         model.addAttribute("currentDate", new java.util.Date());
158         return "view";    // /WEB-INF/views/view + .jsp
159     }
160
161     @RequestMapping("/fruits")
162     public String fruits(Model model){
163         String [] array = {"Apple", "Mango", "Lemon", "Grape"};
164
165         model.addAttribute("fruits", array);
166
167         return "fruits";    // /WEB-INF/views/fruits + .jsp
168     }
169 }

```

3. View에 Data 전달

1)src/main/webapp/WEB-INF/views/view.jsp 생성

```

175 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
176 <!DOCTYPE html>
177 <html>
178     <head>
179         <meta charset="UTF-8">
180         <title>Insert title here</title>
181     </head>
182     <body>
183         <h1>view.jsp 입니다.</h1>
184         현재 날짜와 시간은 ${currentDate} 입니다.
185     </body>
186 </html>

```

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

```

190 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
191 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
192 <!DOCTYPE html>
193 <html>
194     <head>
195         <meta charset="UTF-8">
196         <title>Insert title here</title>
197     </head>
198     <body>
199         <h2>fruits.jsp</h2>
200         <ul>과일 종류
201         <c:forEach items="${fruits}" var="fruit">
202             <li>${fruit}</li>
203         </c:forEach>
204     </ul>
205 </body>
206 </html>

```

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

4)http://localhost:8080/demo/fruits --> /fruits.jsp

4. View에 ModelAndView 객체로 data 전달

1)UserController.java에 아래의 코드 추가

```

215 @RequestMapping(value = "/demo", method = RequestMethod.GET)
216 public ModelAndView demo() {
217     /*
218     ModelAndView mav = new ModelAndView("view2");
219     mav.addObject("username", "한지민");
220     mav.addObject("currentDate", new java.util.Date());

```

```

221     return mav;
222     */
223     ModelAndView mav = new ModelAndView();
224     mav.addObject("userid", "example");
225     mav.addObject("passwd", "12345678");
226     mav.setViewName("/demo");
227     return mav;
228 }
229

```

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

```

231
232 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
233 <!DOCTYPE html>
234 <html>
235     <head>
236         <meta charset="UTF-8">
237         <title>Insert title here</title>
238     </head>
239     <body>
240         아이디 : ${userid} <br />
241         패스워드 : ${passwd}
242     </body>
243 </html>
244

```

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

```

246     아이디 : example
247     패스워드 : 12345678
248
249

```

5. Controller class에 @RequestMapping 적용

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

```

252
253     package com.example.biz;
254
255     import org.springframework.stereotype.Controller;
256     import org.springframework.web.bind.annotation.RequestMapping;
257     import org.springframework.web.bind.annotation.RequestMethod;
258     import org.springframework.web.servlet.ModelAndView;
259
260     @Controller
261     @RequestMapping("/bbs")
262     public class StudentController {
263
264         @RequestMapping(value="/get", method = RequestMethod.GET)
265         public ModelAndView getStudent() {
266
267             ModelAndView mav = new ModelAndView();
268             mav.setViewName("/bbs/get");
269             mav.addObject("name", "한지민");
270             mav.addObject("age", 25);
271             return mav;
272         }
273     }
274

```

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

```

276 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
277 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
278 "http://www.w3.org/TR/html4/loose.dtd">
279 <html>
280 <head>
281     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
282     <title>Insert title here</title>
283 </head>
284 <body>
285     학생 이름 : ${name} <br />
286     학생 나이 : ${age}
287 </body>
288 </html>
289

```

3)http://localhost:8080/demo/bbs/get

```

290     학생 이름 : 한지민
291     학생 나이 : 25
292
293

```

```

294 -----
295 Task4. 다양한 GET Request 처리하기
296 1. Package Explorer > right-click > New > Spring Legacy Project
297 2. Select Spring MVC Project
298 3. Project name : MVCDemo > Next
299 4. Enter a topLevelPackage : com.example.biz > Finish
300
301 5. pom.xml 수정하기
302     <properties>
303         <java-version>11</java-version>
304         <org.springframework-version>5.3.12</org.springframework-version>
305         <org.aspectj-version>1.9.7</org.aspectj-version>
306         <org.slf4j-version>1.7.32</org.slf4j-version>
307     </properties>
308     ...
309     <dependency>
310         <groupId>javax.servlet</groupId>
311         <artifactId>javax.servlet-api</artifactId>
312         <version>4.0.1</version>
313         <scope>provided</scope>
314     </dependency>
315     <dependency>
316         <groupId>javax.servlet.jsp</groupId>
317         <artifactId>javax.servlet.jsp-api</artifactId>
318         <version>2.3.3</version>
319         <scope>provided</scope>
320     </dependency>
321     <dependency>
322         <groupId>org.junit.jupiter</groupId>
323         <artifactId>junit-jupiter-api</artifactId>
324         <version>5.8.1</version>
325         <scope>test</scope>
326     </dependency>
327     ...
328     <build>
329         <plugins>
330             <plugin>
331                 <artifactId>maven-eclipse-plugin</artifactId>
332                 <version>2.10</version>
333             ...
334             </plugin>
335             <plugin>
336                 <groupId>org.apache.maven.plugins</groupId>
337                 <artifactId>maven-compiler-plugin</artifactId>
338                 <version>3.8.1</version>
339                 <configuration>
340                     <source>11</source>
341                     <target>11</target>
342                 ...
343             </plugin>
344             <plugin>
345                 <groupId>org.codehaus.mojo</groupId>
346                 <artifactId>exec-maven-plugin</artifactId>
347                 <version>3.0.0</version>
348             ...
349
350 6. pom.xml > right-click > Run As > Maven install
351     [INFO] BUILD SUCCESS
352
353
354 7. Project > right-click > Properties > Project Facets > Select Java > Change Version 11
355     -Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close
356
357
358 8. src/main/java/com.example.biz/RequestController.java 생성
359
360     package com.example.biz;
361
362     import org.springframework.stereotype.Controller;
363
364     @Controller
365     public class RequestController {
366
367

```

368 9. HttpServletRequest class 이용하기
369 1)RequestController.java

```
370  
371 @RequestMapping(value="/confirm", method=RequestMethod.GET)  
372 public String confirm(HttpServletRequest request, Model model) {  
373     String userid = request.getParameter("userid");  
374     String passwd = request.getParameter("passwd");  
375     String name = request.getParameter("name");  
376     int age = Integer.parseInt(request.getParameter("age"));  
377     String gender = request.getParameter("gender");  
378  
379     model.addAttribute("userid", userid);  
380     model.addAttribute("passwd", passwd);  
381     model.addAttribute("name", name);  
382     model.addAttribute("age", age);  
383     model.addAttribute("gender", gender);  
384     return "confirm";  
385 }
```

386
387 2)src/main/webapp/WEB-INF/views/confirm.jsp

```
388  
389 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  
390 <!DOCTYPE html>  
391 <html>  
392     <head>  
393         <meta charset="UTF-8">  
394         <title>Insert title here</title>  
395     </head>  
396     <body>  
397         아이디 : ${userid} <br />  
398         패스워드 : ${passwd} <br />  
399         사용자 이름 : ${name} <br />  
400         나이 : ${age} <br />  
401         성별 : ${gender} <br />  
402     </body>  
403 </html>
```

404
405 3)Project right-click > Run As > Run on Server > restart

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

```
407     아이디 : jimin  
408     패스워드 : 1234  
409     사용자 이름 : 한지민  
410     나이 : 25  
411     성별 : 여성
```

412
413
414 10. @RequestParam Annotation 이용하기

415 1)RequestController.java

```
416 @RequestMapping(value="/confirm", method=RequestMethod.GET)  
417 public String confirm(@RequestParam("userid") String userid,  
418                       @RequestParam("passwd") String passwd,  
419                       @RequestParam("name") String name,  
420                       @RequestParam("age") int age,  
421                       @RequestParam("gender") String gender ,Model model) {  
422  
423     model.addAttribute("userid", userid);  
424     model.addAttribute("passwd", passwd);  
425     model.addAttribute("name", name);  
426     model.addAttribute("age", age);  
427     model.addAttribute("gender", gender);  
428     return "confirm";  
429 }
```

430
431 2)src/main/webapp/WEB-INF/views/confirm.jsp

```
432  
433 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  
434 <!DOCTYPE html>  
435 <html>  
436     <head>  
437         <meta charset="UTF-8">  
438         <title>Insert title here</title>  
439     </head>  
440     <body>  
441         아이디 : ${userid} <br />
```

```

442         패스워드 : ${passwd} <br />
443         사용자 이름 : ${name} <br />
444         나이 : ${age} <br />
445         성별 : ${gender} <br />
446     </body>
447 </html>

```

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

```

450 아이디 : jimin
451 패스워드 : 1234
452 사용자 이름 : 한지민
453 나이 : 25
454 성별 : 여성

```

11. Data Commander 객체 사용하기1

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

```

460 package com.example.vo;
461
462 public class UserVO {
463     private String userid;
464     private String passwd;
465     private String name;
466     private int age;
467     private String gender;
468
469     public UserVO(){}
470     public UserVO(String userid, String passwd, String name, int age, String gender){
471         this.userid = userid;
472         this.passwd = passwd;
473         this.name = name;
474         this.age = age;
475         this.gender = gender;
476     }
477     public String getUserid() {
478         return userid;
479     }
480     public void setUserid(String userid) {
481         this.userid = userid;
482     }
483     public String getPasswd() {
484         return passwd;
485     }
486     public void setPasswd(String passwd) {
487         this.passwd = passwd;
488     }
489     public String getName() {
490         return name;
491     }
492     public void setName(String name) {
493         this.name = name;
494     }
495     public int getAge() {
496         return age;
497     }
498     public void setAge(int age) {
499         this.age = age;
500     }
501     public String getGender() {
502         return gender;
503     }
504     public void setGender(String gender) {
505         this.gender = gender;
506     }
507     @Override
508     public String toString() {
509         return "UserVO [userid=" + userid + ", passwd=" + passwd + ", name=" + name + ", age=" + age +
510             ", gender="
511             + gender + "]);
512     }

```

2)RequestController.java

```

515 @RequestMapping(value="/confirm", method=RequestMethod.GET)
516 public String confirm(@RequestParam("userid") String userid,
517     @RequestParam("passwd") String passwd,
518     @RequestParam("name") String name,
519     @RequestParam("age") int age,
520     @RequestParam("gender") String gender, Model model) {
521
522     UserVO userVO = new UserVO();
523     userVO.setUserId(userid);
524     userVO.setPasswd(passwd);
525     userVO.setName(name);
526     userVO.setAge(age);
527     userVO.setGender(gender);
528
529     model.addAttribute("userVO", userVO);
530
531     return "confirm1";
532 }
533

```

3)src/main/webapp/WEB-INF/views/confirm1.jsp

```

537 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
538 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
539 <c:set var="user" value="${userVO}"/>
540 <!DOCTYPE html>
541 <html>
542 <head>
543 <meta charset="UTF-8">
544 <title>Insert title here</title>
545 </head>
546 <body>
547 <h1>confirm1.jsp</h1>
548 <h2>사용자 정보</h2>
549 아이디 : ${user.userid} <br />
550 패스워드 : ${user.passwd} <br />
551 이름 : ${user.name} <br />
552 나이 : ${user.age} <br />
553 성별 : ${user.gender}
554 </body>
555 </html>
556

```

4)localhost:8080/biz/confirm?name=한지민&gender=여성&age=25&userid=jimin&passwd=1234 confirm1.jsp

```

560 사용자 정보
561
562 아이디 : jimin
563 패스워드 : 1234
564 사용자 이름 : 한지민
565 나이 : 25
566 성별 : 여성
567

```

12. Data Commander 객체 이용하기2

1)RequestController.java

```

572 @RequestMapping(value="/confirm", method=RequestMethod.GET)
573 public String confirm(UserVO userVO) {
574
575     return "confirm2";
576 }
577

```

2)src/main/webapp/WEB-INF/views/confirm2.jsp

```

580 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
581 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
582 <c:set var="user" value="${userVO}"/>
583 <!DOCTYPE html>
584 <html>
585 <head>
586 <meta charset="UTF-8">
587 <title>Insert title here</title>
588 </head>

```



```

589     <body>
590         <h1>confirm2.jsp</h1>
591         <h2>사용자 정보</h2>
592         아이디 : ${user.userid} <br />
593         패스워드 : ${user.passwd} <br />
594         이름 : ${user.name} <br />
595         나이 : ${user.age} <br />
596         성별 : ${user.gender}
597     </body>
598 </html>
599

```

3)localhost:8080/biz/confirm?name=한지민&gender=여성&age=25&userid=jimin&passwd=1234
confirm2.jsp

```

603     사용자 정보
604
605     아이디 : jimin
606     패스워드 : 1234
607     사용자 이름 : 한지민
608     나이 : 25
609     성별 : 여성
610
611

```

13. @PathVariable 이용하기

1)RequestController.java

```

615     @RequestMapping(value="/confirm/{userid}/{passwd}/{name}/{age}/{gender}",
616                     method=RequestMethod.GET)
617     public String confirm(@PathVariable String userid, @PathVariable String passwd,
618                          @PathVariable String name, @PathVariable int age,
619                          @PathVariable String gender, Model model) {
620         model.addAttribute("userInfo", new UserVO(userid, passwd, name, age, gender));
621         return "confirm3";
622     }
623

```

2)src/main/webapp/WEB-INF/views/confirm3.jsp

```

625     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
626     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
627     <c:set var="user" value="${userInfo}" />
628     <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
629         "http://www.w3.org/TR/html4/loose.dtd">
630     <html>
631     <head>
632     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
633     <title>Insert title here</title>
634     </head>
635     <body>
636         <h1>confirm3.jsp</h1>
637         <h2>사용자 정보</h2>
638         아이디 : ${user.userid} <br />
639         패스워드 : ${user.passwd} <br />
640         이름 : ${user.name} <br />
641         나이 : ${user.age} <br />
642         성별 : ${user.gender}
643     </body>
644 </html>
645

```

3)localhost:8080/biz/confirm/jimin/1234/한지민/25/여성
confirm3.jsp

```

648     사용자 정보
649
650     아이디 : jimin
651     패스워드 : 1234
652     사용자 이름 : 한지민
653     나이 : 25
654     성별 : 여성
655
656

```

Task5. @RequestMapping Parameter 다루기

1. GET 방식과 POST 방식

1)src/main/java/com.example.biz/HomeController.java

```
@RequestMapping(value="/login", method=RequestMethod.POST)
public String login(@RequestParam("userid") String userid,
    @RequestParam("passwd") String passwd,
    Model model) {

    model.addAttribute("userid", userid);
    model.addAttribute("passwd", passwd);
    return "login";
}
```

2)src/main/webapp/resources/login.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>로그인 폼</title>
</head>
<body>
    <form method="GET" action="/biz/login">
        아이디 : <input type="text" name="userid" /><br />
        패스워드 : <input type="password" name="passwd" /><br />
        <input type="submit" value="로그인하기" />
    </form>
</body>
</html>
```

3)http://localhost:8080/biz/resources/login.html에서 submit 하면 405 error 발생

4)왜냐하면 서로의 method가 불일치하기 때문

5)해결방법

-src/main/java/com.example.biz/HomeController.java 수정

-즉 login method(요청 처리 method)의 이름은 같지만 parameter의 type과 return type이 틀리기 때문에 Method Overloading 됨.

```
@RequestMapping(value="/login", method=RequestMethod.POST)
public String login(@RequestParam("userid") String userid,
    @RequestParam("passwd") String passwd,
    Model model) {

    model.addAttribute("userid", userid);
    model.addAttribute("passwd", passwd);
    return "login";
}

@RequestMapping(value="/login", method=RequestMethod.GET)
public ModelAndView login(@RequestParam("userid") String userid,
    @RequestParam("passwd") String passwd) {

    ModelAndView mav = new ModelAndView();
    mav.addObject("userid", userid);
    mav.addObject("passwd", passwd);
    mav.setViewName("login");
    return mav;
}
```

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

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    아이디 : ${userid} <br />
    패스워드 : ${passwd}
</body>
</html>
```

7)http://localhost:8080/biz/resources/login.html

아이디 : jimin
패스워드 : 1234

```
807 9) http://localhost:8080/biz/register
```

808 사용자 정보
809
810 아이디 : jimin
811 패스워드 : 1234
812 사용자 이름 : 한지민
813 나이 : 25
814 성별 : 여성
815
816

817 3. redirect: 키워드 이용하기

818 1)src/main/java/com.example.biz/HomeController.java

```
819
820 @RequestMapping("/verify")
821 public String verify(HttpServletRequest request, Model model) {
822     String userid = request.getParameter("userid");
823     if(userid.equals("admin")) { //만일 userid가 admin 이면 /admin으로 리다이렉트
824         return "redirect:admin";
825     }
826     return "redirect:user"; //만일 userid가 admin 이 아니면 /user로 리다이렉트
827     //return "redirect:http://www.naver.com"; //절대 경로도 가능
828 }
829
```

```
830 @RequestMapping("/admin")
831 public String verify1(Model model) {
832     model.addAttribute("authority", "관리자권한");
833     return "admin";
834 }
835
```

```
836 @RequestMapping("/user")
837 public String verify2(Model model) {
838     model.addAttribute("authority", "일반사용자");
839     return "user";
840 }
841
```

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

```
843 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
844 <!DOCTYPE html>
845 <html>
846 <head>
847 <meta charset=UTF-8">
848 <title>Insert title here</title>
849 </head>
850 <body>
851 <h1>관리자 페이지</h1>
852 권한 : ${authority}
853 </body>
854 </html>
855
```

856 3)/src/main/webapp/WEB-INF/views/user.jsp

```
857
858 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
859 <!DOCTYPE html>
860 <html>
861 <head>
862 <meta charset=UTF-8">
863 <title>Insert title here</title>
864 </head>
865 <body>
866 <h1>일반 사용자 페이지</h1>
867 권한 : ${authority}
868 </body>
869 </html>
870
```

871 4)http://localhost:8080/biz/verify?userid=admin --> http://localhost:8080/biz/admin

872 5)http://localhost:8080/biz/verify?userid=user --> https://www.naver.com

873
874

875 -----

876 Task6. Database와 연동하기

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

878 2. Select Spring MVC Project

879 3. Project name : MVCDemo1 > Next

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

881 5. pom.xml 수정하기

```

882 <properties>
883   <java-version>11</java-version>
884   <org.springframework-version>5.3.12</org.springframework-version>
885   <org.aspectj-version>1.9.7</org.aspectj-version>
886   <org.slf4j-version>1.7.32</org.slf4j-version>
887 </properties>
888 ...
889 <dependency>
890   <groupId>javax.servlet</groupId>
891   <artifactId>javax.servlet-api</artifactId>
892   <version>4.0.1</version>
893   <scope>provided</scope>
894 </dependency>
895 <dependency>
896   <groupId>javax.servlet.jsp</groupId>
897   <artifactId>javax.servlet.jsp-api</artifactId>
898   <version>2.3.3</version>
899   <scope>provided</scope>
900 </dependency>
901 <dependency>
902   <groupId>org.junit.jupiter</groupId>
903   <artifactId>junit-jupiter-api</artifactId>
904   <version>5.8.1</version>
905   <scope>test</scope>
906 </dependency>
907 ...
908 <build>
909   <plugins>
910     <plugin>
911       <artifactId>maven-eclipse-plugin</artifactId>
912       <version>2.10</version>
913     ...
914   </plugin>
915   <plugin>
916     <groupId>org.apache.maven.plugins</groupId>
917     <artifactId>maven-compiler-plugin</artifactId>
918     <version>3.8.1</version>
919     <configuration>
920       <source>11</source>
921       <target>11</target>
922     ...
923   </plugin>
924   <plugin>
925     <groupId>org.codehaus.mojo</groupId>
926     <artifactId>exec-maven-plugin</artifactId>
927     <version>3.0.0</version>
928   </plugin>
929 </build>

```

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

931 [INFO] BUILD SUCCESS

932 7. MVCDemo1 Project > right-click > Properties > Project Facets > Select Java > Change Version 11
 933 Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close

936 8. Create Table in MariaDB

```

938 CREATE TABLE Member
939 (
940   userid      VARCHAR(20),
941   username    VARCHAR(20) NOT NULL,
942   userage     TINYINT  NOT NULL,
943   gender      VARCHAR(10) NOT NULL,
944   city        VARCHAR(50),
945   CONSTRAINT member_userid_pk PRIMARY KEY(userid)
946 );

```

947 -반드시 [test] Database의 조합을 utf8_general_ci로 맞추는 것

948 -반드시 Member Table의 기본조합이 utf8_general_ci 임을 확인할 것

950 9. src/main/webapp/static folder 생성

951 1)src/main/webapp/static/css folder

952 2)src/main/webapp/static/images folder

```

956 3)src/main/webapp/static/js folder
957 -jquery-1.12.4.js
958
959 4)src/main/webapp/static/register.html
960 <!DOCTYPE html>
961 <html lang="en">
962 <head>
963     <meta charset="UTF-8">
964     <title>회원 가입</title>
965 </head>
966 <body>
967     <h1>회원 가입 창</h1>
968     <form action="/biz/create" method="post">
969         <ul>
970             <li>ID : <input type="text" name="userid" /></li>
971             <li>이름 : <input type="text" name="username" /></li>
972             <li>나이 : <input type="number" name="age" /></li>
973             <li>성별 : <input type="radio" name="gender" value="남성"/>남성
974                     <input type="radio" name="gender" value="여성"/>여성</li>
975             <li>거주지 : <input type="text" name="city" /></li>
976             <li><input type="submit" value="가입하기" /></li>
977         </ul>
978     </form>
979 </body>
980 </html>
981
982
983 10. src/main/webapp/WEB-INF/spring/appServlet/sevlet-context.xml 수정
984 <resources mapping="/static/**" location="/static/" /> 추가
985
986 <context:component-scan base-package="com.example" /> 수정
987
988
989 11. src/main/resources/mariadb.properties
990 db.driverClass=org.mariadb.jdbc.Driver
991 db.url=jdbc:mariadb://localhost:3306/test
992 db.username=root
993 db.password=javamariadb
994
995
996 12. Spring JDBC 설치
997 1)JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
998
999 <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
1000 <dependency>
1001     <groupId>org.springframework</groupId>
1002     <artifactId>spring-jdbc</artifactId>
1003     <version>5.3.12</version>
1004 </dependency>
1005
1006 2)pom.xml에 붙여 넣고 Maven Install 하기
1007 [INFO] BUILD SUCCESS
1008
1009
1010 13. MariaDB Jdbc Driver library 검색 및 설치
1011 1)Maven Repository 에서 'mariadb'로 검색하여 MariaDB Java Client를 설치한다.
1012
1013 <dependency>
1014     <groupId>org.mariadb.jdbc</groupId>
1015     <artifactId>mariadb-java-client</artifactId>
1016     <version>2.7.4</version>
1017 </dependency>
1018
1019 2)pom.xml에 붙여 넣고 Maven Install 하기
1020 [INFO] BUILD SUCCESS
1021
1022
1023 14. src/main/webapp/WEB-INF/spring/root-context.xml
1024 <?xml version="1.0" encoding="UTF-8"?>
1025 <beans xmlns="http://www.springframework.org/schema/beans"
1026     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1027     xmlns:context="http://www.springframework.org/schema/context"
1028     xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd

```

```

1029     http://www.springframework.org/schema/context
1030     http://www.springframework.org/schema/context/spring-context-4.3.xsd">
1031
1032     <!-- Root Context: defines shared resources visible to all other web components -->
1033     <!-- <context:property-placeholder location="classpath:mariadb.properties"/> -->
1034
1035     <!-- or -->
1036     <bean id="my.propertyConfigurer"
1037           class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
1038         <property name="locations">
1039             <list>
1040                 <value>classpath:dbinfo.properties</value>
1041             </list>
1042         </property>
1043     </bean>
1044
1045     <bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource">
1046         <property name="driverClassName" value="{db.driverClass}"/>
1047         <property name="url" value="{db.url}"/>
1048         <property name="username" value="{db.username}" />
1049         <property name="password" value="{db.password}" />
1050     </bean>
1051
1052     <!-- <bean id="dataSource" class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
1053         <property name="driverClass" value="{db.driverClass}" />
1054         <property name="url" value="{db.url}" />
1055         <property name="username" value="{db.username}" />
1056         <property name="password" value="{db.password}" />
1057     </bean> -->
1058
1059     <!-- or -->
1060     <bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource" destroy-method="close">
1061         <constructor-arg>
1062             <bean class="com.zaxxer.hikari.HikariConfig">
1063                 <constructor-arg>
1064                     <props>
1065                         <prop key="jdbcUrl">{db.url}</prop>
1066                         <prop key="username">{db.username}</prop>
1067                         <prop key="password">{db.password}</prop>
1068                     </props>
1069                 </constructor-arg>
1070                 <property name="driverClassName" value="{db.driverClass}"/>
1071                 <property name="minimumIdle" value="5" />
1072                 <property name="maximumPoolSize" value="10" />
1073                 <property name="connectionTestQuery" value="select 1 from sys.dual" />
1074                 <property name="connectionTimeout" value="300000" />
1075             </bean>
1076         </constructor-arg>
1077     </bean>
1078
1079     <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
1080         <property name="dataSource" ref="dataSource" />
1081     </bean>
1082 </beans>

```

15. src/test/java/com.example.biz/TestApp class 생성

```

1084 1)com.example.biz > right-click > New > JUnit Test Case
1085 2)Select [New JUnit 4 test]
1086 3)Name : TestApp
1087 4)Finish
1088     package com.example.biz;
1089
1090     import org.junit.Before;
1091     import org.junit.Test;
1092     import org.springframework.context.ApplicationContext;
1093     import org.springframework.context.support.GenericXmlApplicationContext;
1094     import org.springframework.jdbc.core.JdbcTemplate;
1095
1096     public class TestApp {
1097         private ApplicationContext ctx;
1098
1099         @Before
1100         public void init() {

```

```

1101         this.ctx = new
1102             GenericXmlApplicationContext("file:src/main/webapp/WEB-INF/spring**/root-context.xml");
1103     }
1104     @Test
1105     public void test() {
1106         JdbcTemplate jdbcTemplate = this.ctx.getBean("jdbcTemplate", JdbcTemplate.class);
1107         System.out.println(jdbcTemplate);
1108     }
1109 }

```

5)Run as > JUnit Test > Green bar

1111
1112
1113 16. package 생성

- 1114 1)src/main/java/com.example.vo
- 1115 2)src/main/java/com.example.dao
- 1116 3)src/main/java/com.example.service

1117
1118
1119 17. src/com.example.vo.MemberVO class 생성

```

1120
1121 package com.example.vo;
1122
1123 public class MemberVO {
1124     private String userid;
1125     private String username;
1126     private int age;
1127     private String gender;
1128     private String city;
1129
1130     public MemberVO() {}
1131
1132     public MemberVO(String userid, String username, int age, String gender, String city) {
1133         this.userid = userid;
1134         this.username = username;
1135         this.age = age;
1136         this.gender = gender;
1137         this.city = city;
1138     }
1139
1140     public String getUserid() {
1141         return userid;
1142     }
1143
1144     public void setUserid(String userid) {
1145         this.userid = userid;
1146     }
1147
1148     public String getUsername() {
1149         return username;
1150     }
1151
1152     public void setUsername(String username) {
1153         this.username = username;
1154     }
1155
1156     public int getAge() {
1157         return age;
1158     }
1159
1160     public void setAge(int age) {
1161         this.age = age;
1162     }
1163
1164     public String getGender() {
1165         return gender;
1166     }
1167
1168     public void setGender(String gender) {
1169         this.gender = gender;
1170     }
1171
1172     public String getCity() {
1173         return city;

```



```

1174     }
1175
1176     public void setCity(String city) {
1177         this.city = city;
1178     }
1179
1180     @Override
1181     public String toString() {
1182         return "MemberVO [userid=" + userid + ", username=" + username + ", age=" + age + ", gender=" +
1183             gender
1184             + ", city=" + city + "]";
1185     }
1186 }
1187
1188 18. com/example/dao
1189 1)MemberDao interface
1190     package com.example.dao;
1191
1192     import java.util.List;
1193
1194     import com.example.vo.MemberVO;
1195
1196     public interface MemberDao {
1197         int create(MemberVO memberVo);
1198         MemberVO read(String userid);
1199         List<MemberVO> readAll();
1200         int update(MemberVO memberVo);
1201         int delete(String userid);
1202     }
1203
1204 2)MemberDaoImpl.java
1205     package com.example.dao;
1206
1207     import java.util.List;
1208
1209     import org.springframework.beans.factory.annotation.Autowired;
1210     import org.springframework.jdbc.core.JdbcTemplate;
1211     import org.springframework.stereotype.Repository;
1212
1213     import com.example.vo.MemberVO;
1214
1215     @Repository("memberDao")
1216     public class MemberDaoImpl implements MemberDao {
1217         @Autowired
1218         JdbcTemplate jdbcTemplate;
1219
1220         @Override
1221         public int create(MemberVO memberVo) {
1222             return 0;
1223         }
1224
1225         @Override
1226         public MemberVO read(String userid) {
1227             return null;
1228         }
1229
1230         @Override
1231         public List<MemberVO> readAll() {
1232             return null;
1233         }
1234
1235         @Override
1236         public int update(MemberVO memberVo) {
1237             return 0;
1238         }
1239
1240         @Override
1241         public int delete(String userid) {
1242             return 0;
1243         }
1244     }
1245
1246

```

```

1247 19. com.example.service
1248     1)MemberService interface
1249         package com.example.service;
1250
1251         import java.util.List;
1252
1253         import com.example.vo.MemberVO;
1254
1255         public interface MemberService {
1256             int create(MemberVO memberVo);
1257             MemberVO read(String userid);
1258             List<MemberVO> readAll();
1259             int update(MemberVO memberVo);
1260             int delete(String userid);
1261         }
1262
1263     2)MemberServiceImpl.java
1264         package com.example.service;
1265
1266         import java.util.List;
1267
1268         import org.springframework.beans.factory.annotation.Autowired;
1269         import org.springframework.stereotype.Service;
1270
1271         import com.example.dao.MemberDao;
1272         import com.example.vo.MemberVO;
1273
1274         @Service("memberService")
1275         public class MemberServiceImpl implements MemberService {
1276             @Autowired
1277             MemberDao memberDao;
1278
1279             @Override
1280             public int create(MemberVO memberVo) {
1281                 return 0;
1282             }
1283
1284             @Override
1285             public MemberVO read(String userid) {
1286                 return null;
1287             }
1288
1289             @Override
1290             public List<MemberVO> readAll() {
1291                 return null;
1292             }
1293
1294             @Override
1295             public int update(MemberVO memberVo) {
1296                 return 0;
1297             }
1298
1299             @Override
1300             public int delete(String userid) {
1301                 return 0;
1302             }
1303         }
1304
1305
1306 20. com.example.biz
1307     1)HomeController.java
1308
1309         package com.example.biz;
1310
1311         import org.springframework.beans.factory.annotation.Autowired;
1312         import org.springframework.stereotype.Controller;
1313
1314         import com.example.service.MemberService;
1315
1316         /**
1317          * Handles requests for the application home page.
1318          */
1319         @Controller
1320         public class HomeController {

```

```
1321     @Autowired
1322     MemberService memberService;
1323 }
1324
1325
```

1326 21. Data Insert

```
1327 1)/src/main/webapp/static/register.html
1328 2)com.example.biz/HomeController.java
1329
```

```
1330     @Controller
1331     public class HomeController {
1332         @Autowired
1333         MemberService memberService;
1334
1335         @RequestMapping(value = "/create", method = RequestMethod.POST)
1336         public String home(MemberVO memberVo, Model model) {
1337             int row = this.memberService.create(memberVo);
1338             if(row == 1) model.addAttribute("status", "Insert Success");
1339             else model.addAttribute("status", "Insert Failure");
1340             return "create";
1341         }
1342     }
1343
```

```
1344 3)com.example.service/MemberServiceImpl.java
1345
```

```
1346     @Service("memberService")
1347     public class MemberServiceImpl implements MemberService {
1348         @Autowired
1349         MemberDao memberDao;
1350
1351         @Override
1352         public int create(MemberVO memberVo) {
1353             return this.memberDao.create(memberVo);
1354         }
1355     }
1356
```

```
1357 4)com.example.dao.MemberDaoImpl.java
1358
```

```
1359     @Repository("memberDao")
1360     public class MemberDaoImpl implements MemberDao {
1361         @Autowired
1362         JdbcTemplate jdbcTemplate;
1363
1364         @Override
1365         public int create(MemberVO memberVo) {
1366             String sql = "INSERT INTO Member VALUES(?,?,?,?)";
1367             return this.jdbcTemplate.update(sql, memberVo.getUserid(),
1368                 memberVo.getUsername(), memberVo.getAge(),
1369                 memberVo.getGender(), memberVo.getCity());
1370         }
1371     }
1372
```

```
1373 5)views/create.jsp
1374
```

```
1375     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
1376     <!DOCTYPE html>
1377     <html>
1378     <head>
1379     <meta charset="UTF-8">
1380     <title>Insert title here</title>
1381     </head>
1382     <body>
1383     <h1>${status}</h1>
1384     </body>
1385     </html>
1386
```

1388 22. POST 발송시 한글 깨짐 처리하기

```
1389 1)web.xml
1390
```

```
1391     <filter>
1392     <filter-name>encodingFilter</filter-name>
1393     <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
1394     <init-param>
```

```
1395         <param-name>encoding</param-name>
1396         <param-value>UTF-8</param-value>
1397     </init-param>
1398 </filter>
1399 <filter-mapping>
1400     <filter-name>encodingFilter</filter-name>
1401     <url-pattern>/*</url-pattern>
1402 </filter-mapping>
```

1403
1404

1405 23. Test

```
1406 1)http://localhost:8080/biz/static/register.html
1407 2)http://localhost:8080/biz/create
1408     Insert Success
```

1409
1410

1411 24. Data Select

1412 1)HomeController.java

```
1413
1414     @RequestMapping(value = "/view/{userid}", method = RequestMethod.GET)
1415     public String view(@PathVariable String userid, Model model) {
1416         MemberVO memberVo = this.memberService.read(userid);
1417         model.addAttribute("member", memberVo);
1418         return "view";
1419     }
1420
```

1421

1422 2)MemberServiceImpl.java

```
1423
1424     @Override
1425     public MemberVO read(String userid) {
1426         return this.memberDao.read(userid);
1427     }
1428
```

1429

1430 3)MemberDaoImpl.java

```
1431
1432     @Override
1433     public MemberVO read(String userid) {
1434         String sql = "SELECT * FROM Member WHERE userid = ?";
1435         return this.jdbcTemplate.queryForObject(sql, new Object[] {userid},
1436             new MyRowMapper());
1437     }
1438
1439     class MyRowMapper implements RowMapper<MemberVO>{
1440         @Override
1441         public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1442             MemberVO memberVo = new MemberVO(rs.getString("userid"),
1443                 rs.getString("username"), rs.getInt("userage"),
1444                 rs.getString("gender"), rs.getString("city"));
1445             return memberVo;
1446         }
1447     }
1448
```

1449

1450 4)views/view.jsp

```
1451
1452     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
1453     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
1454     <c:set var="user" value="${member}" />
1455     <!DOCTYPE html>
1456     <html>
1457     <head>
1458         <meta charset="UTF-8">
1459         <title>Insert title here</title>
1460         <script src="/biz/static/js/jquery-1.12.4.js"></script>
1461         <script>
1462             $(function(){
1463                 $("#btnList").bind("click", function(){
1464                     location.href = "/biz/list";
1465                 });
1466                 $("#btnDelete").bind("click", function(){
1467                     location.href = "/biz/delete/${user.userid}";
1468                 });
1469             });
1470         </script>
1471     </head>
```

```

1469 <body>
1470   <h1>${user.username}의정보</h1>
1471   <form action="/biz/update" method="post">
1472     <input type="hidden" name="userid" value = "${user.userid}" />
1473     <ul>
1474       <li>아이디 : ${user.userid }</li>
1475       <li>나이 : <input type='number' name="age" value='${user.age}' /></li>
1476       <li>성별 : <c:if test='${user.gender eq "남성"}'>
1477         <input type="radio" name="gender" value="남성" checked />남성&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
1478         <input type="radio" name="gender" value="여성" />여성
1479       </c:if>
1480       <c:if test='${user.gender eq "여성"}'>
1481         <input type="radio" name="gender" value="남성" />남성&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
1482         <input type="radio" name="gender" value="여성" checked />여성
1483       </c:if>
1484     </li>
1485     <li>거주지 : <input type="text" name="city" value="${user.city }" /></li>
1486     <li><input type='submit' value='수정하기' /></li>
1487     <li><input type='button' value='삭제하기' id="btnDelete"/></li>
1488     <li><input type='button' value='목록으로' id="btnList"/></li>
1489   </ul>
1490 </form>
1491 </body>
1492 </html>

```

5) Test

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

25. Data List

1)HomeController.java

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

2)MemberServiceImpl.java

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

3)MemberDaoImpl.java

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

4)iews/list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>Member List</h1>
    <table border='1'>
        <thead>
            <tr>
                <th>아이디</th><th>이름</th><th>나이</th><th>성별</th><th>거주지</th>
            </tr>
        </thead>
        <tbody>
            <c:forEach items="${userlist}" var="user">
```

```

1543         <tr>
1544             <td><a
1545                 href="/biz/view/${user.userid}">${user.userid}</a></td><td>${user.username}</td>
1546             <td>${user.age}</td><td>${user.gender}</td>
1547             <td>${user.city}</td>
1548         </tr>
1549     </c:forEach>
1550 </tbody>
1551 </table>
1552 </body>
1553 </html>

```

5)Test

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

26. Data Delete

1)HomeController.java

```

1561 @RequestMapping(value = "/delete/{userid}", method = RequestMethod.GET)
1562 public String delete(@PathVariable String userid) {
1563     this.memberService.delete(userid);
1564     return "redirect:/list";
1565 }

```

2)MemberServiceImpl.java

```

1569 @Override
1570 public int delete(String userid) {
1571     return this.memberDao.delete(userid);
1572 }

```

3)MemberDaoImpl.java

```

1576 @Override
1577 public int delete(String userid) {
1578     String sql = "DELETE FROM Member WHERE userid = ?";
1579     return this.jdbcTemplate.update(sql, userid);
1580 }

```

4)Test

<http://localhost:8080/biz/delete/chulsu>

27. Data Update

1)HomeController.java

```

1589 @RequestMapping(value = "/update", method = RequestMethod.POST)
1590 public String update(@RequestParam("userid") String userid,
1591                     @RequestParam("age") int age,
1592                     @RequestParam("gender") String gender,
1593                     @RequestParam("city") String city) {
1594     this.memberService.update(
1595         new MemberVO(userid, "", age, gender, city));
1596     return "redirect:/list";
1597 }

```

2)MemberServiceImpl.java

```

1601 @Override
1602 public int update(MemberVO memberVo) {
1603     return this.memberDao.update(memberVo);
1604 }

```

3)MemberDaoImpl.java

```

1608 @Override
1609 public int update(MemberVO memberVo) {
1610     String sql = "UPDATE Member SET uselage = ?, gender = ?, city = ? " +
1611                 "WHERE userid = ?";
1612     return this.jdbcTemplate.update(sql, memberVo.getAge(),
1613                                     memberVo.getGender(), memberVo.getCity(), memberVo.getUserid());
1614 }

```

```
1616 4)Test
1617    http://localhost:8080/biz/list에서
1618    해당 ID Click
1619    데이터 수정
1620    [수정하기] button click
1621
1622
```

1623 28. All Codes

1624 1)HomeController.java

```
1625
1626    package com.example.biz;
1627
1628    import java.util.List;
1629
1630    import org.springframework.beans.factory.annotation.Autowired;
1631    import org.springframework.stereotype.Controller;
1632    import org.springframework.ui.Model;
1633    import org.springframework.web.bind.annotation.PathVariable;
1634    import org.springframework.web.bind.annotation.RequestMapping;
1635    import org.springframework.web.bind.annotation.RequestMethod;
1636    import org.springframework.web.bind.annotation.RequestParam;
1637
1638    import com.example.service.MemberService;
1639    import com.example.vo.MemberVO;
1640
1641    /**
1642     * Handles requests for the application home page.
1643     */
1644    @Controller
1645    public class HomeController {
1646        @Autowired
1647        MemberService memberService;
1648
1649        @RequestMapping(value = "/create", method = RequestMethod.POST)
1650        public String home(MemberVO memberVo, Model model) {
1651            int row = this.memberService.create(memberVo);
1652            if(row == 1) model.addAttribute("status", "Insert Success");
1653            else model.addAttribute("status", "Insert Failure");
1654            return "create";    // /WEB-INF/views/create.jsp
1655        }
1656
1657        @RequestMapping(value = "/view/{userid}", method = RequestMethod.GET)
1658        public String view(@PathVariable String userid, Model model) {
1659            MemberVO memberVo = this.memberService.read(userid);
1660            model.addAttribute("member", memberVo);
1661            return "view";
1662        }
1663
1664        @RequestMapping(value = "/list", method = RequestMethod.GET)
1665        public String list(Model model) {
1666            List<MemberVO> list = this.memberService.readAll();
1667            model.addAttribute("userlist", list);
1668            return "list";    // /WEB-INF/views/list.jsp
1669        }
1670
1671        @RequestMapping(value = "/delete/{userid}", method = RequestMethod.GET)
1672        public String delete(@PathVariable String userid) {
1673            this.memberService.delete(userid);
1674            return "redirect:/list";
1675        }
1676
1677        @RequestMapping(value = "/update", method = RequestMethod.POST)
1678        public String update(@RequestParam("userid") String userid,
1679                             @RequestParam("age") int age,
1680                             @RequestParam("gender") String gender,
1681                             @RequestParam("city") String city) {
1682            this.memberService.update(
1683                new MemberVO(userid, "", age, gender, city));
1684            return "redirect:/list";
1685        }
1686    }
1687
```

1688 2)MemberServiceImpl.java

1689

```

1690 package com.example.service;
1691
1692 import java.util.List;
1693
1694 import org.springframework.beans.factory.annotation.Autowired;
1695 import org.springframework.stereotype.Service;
1696
1697 import com.example.dao.MemberDao;
1698 import com.example.vo.MemberVO;
1699
1700 @Service("memberService")
1701 public class MemberServiceImpl implements MemberService {
1702     @Autowired
1703     MemberDao memberDao;
1704
1705     @Override
1706     public int create(MemberVO memberVo) {
1707         return this.memberDao.create(memberVo);
1708     }
1709
1710     @Override
1711     public MemberVO read(String userid) {
1712         return this.memberDao.read(userid);
1713     }
1714
1715     @Override
1716     public List<MemberVO> readAll() {
1717         return this.memberDao.readAll();
1718     }
1719
1720     @Override
1721     public int update(MemberVO memberVo) {
1722         return this.memberDao.update(memberVo);
1723     }
1724
1725     @Override
1726     public int delete(String userid) {
1727         return this.memberDao.delete(userid);
1728     }
1729 }

```

3)MemberDaoImpl.java

```

1731 package com.example.dao;
1732
1733 import java.sql.ResultSet;
1734 import java.sql.SQLException;
1735 import java.util.List;
1736
1737 import org.springframework.beans.factory.annotation.Autowired;
1738 import org.springframework.jdbc.core.JdbcTemplate;
1739 import org.springframework.jdbc.core.RowMapper;
1740 import org.springframework.stereotype.Repository;
1741
1742 import com.example.vo.MemberVO;
1743
1744 @Repository("memberDao")
1745 public class MemberDaoImpl implements MemberDao {
1746     @Autowired
1747     JdbcTemplate jdbcTemplate;
1748
1749     @Override
1750     public int create(MemberVO memberVo) {
1751         String sql = "INSERT INTO Member VALUES(?,?,?,?)";
1752         return this.jdbcTemplate.update(sql, memberVo.getUserid(), memberVo.getUsername(),
1753             memberVo.getAge(),
1754             memberVo.getGender(), memberVo.getCity());
1755     }
1756
1757     class MyRowMapper implements RowMapper<MemberVO> {
1758         @Override
1759         public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1760             MemberVO memberVo = new MemberVO(rs.getString("userid"), rs.getString("username"),
1761                 rs.getInt("usage"),

```



```

1762         rs.getString("gender"), rs.getString("city"));
1763     return memberVo;
1764 }
1765 }
1766
1767 @Override
1768 public MemberVO read(String userid) {
1769     String sql = "SELECT * FROM Member WHERE userid = ?";
1770     return this.jdbcTemplate.queryForObject(sql, new Object[] { userid }, new MyRowMapper());
1771 }
1772
1773 @Override
1774 public List<MemberVO> readAll() {
1775     String sql = "SELECT * FROM Member ORDER BY userid DESC";
1776     return this.jdbcTemplate.query(sql, new MyRowMapper());
1777 }
1778
1779 @Override
1780 public int update(MemberVO memberVo) {
1781     String sql = "UPDATE Member SET usagerage = ?, gender = ?, city = ? " + "WHERE userid = ?";
1782     return this.jdbcTemplate.update(sql, memberVo.getAge(), memberVo.getGender(),
        memberVo.getCity(),
        memberVo.getUserid());
1783 }
1784
1785 @Override
1786 public int delete(String userid) {
1787     String sql = "DELETE FROM Member WHERE userid = ?";
1788     return this.jdbcTemplate.update(sql, userid);
1789 }
1790 }
1791 }

```

```

1792
1793
1794
1795 -----
1796 Task7. Form Data Validation
1797 1. Package Explorer > right-click > New > Other > Spring > Spring Legacy Project
1798 2. Select Spring MVC Project
1799 3. Project name : FormValidationDemo > Next
1800 4. Enter a topLevelPackage : com.example.biz > Finish
1801 5. pom.xml 수정하기
1802 <properties>
1803     <java-version>11</java-version>
1804     <org.springframework-version>5.3.12</org.springframework-version>
1805     <org.aspectj-version>1.9.7</org.aspectj-version>
1806     <org.slf4j-version>1.7.32</org.slf4j-version>
1807 </properties>
1808 ...
1809 <dependency>
1810     <groupId>javax.servlet</groupId>
1811     <artifactId>javax.servlet-api</artifactId>
1812     <version>4.0.1</version>
1813     <scope>provided</scope>
1814 </dependency>
1815 <dependency>
1816     <groupId>javax.servlet.jsp</groupId>
1817     <artifactId>javax.servlet.jsp-api</artifactId>
1818     <version>2.3.3</version>
1819     <scope>provided</scope>
1820 </dependency>
1821 <dependency>
1822     <groupId>org.junit.jupiter</groupId>
1823     <artifactId>junit-jupiter-api</artifactId>
1824     <version>5.8.1</version>
1825     <scope>test</scope>
1826 </dependency>
1827 ...
1828 <build>
1829     <plugins>
1830         <plugin>
1831             <artifactId>maven-eclipse-plugin</artifactId>
1832             <version>2.10</version>
1833         ...
1834     </plugin>

```

```

1835     <plugin>
1836         <groupId>org.apache.maven.plugins</groupId>
1837         <artifactId>maven-compiler-plugin</artifactId>
1838         <version>3.8.1</version>
1839         <configuration>
1840             <source>11</source>
1841             <target>11</target>
1842         ...
1843     </plugin>
1844     <plugin>
1845         <groupId>org.codehaus.mojo</groupId>
1846         <artifactId>exec-maven-plugin</artifactId>
1847         <version>3.0.0</version>
1848
1849

```

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

[INFO] BUILD SUCCESS

7. FormValidationDemo Project > right-click > Properties > Project Facets > Select Java > Change Version 11 Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close

8. UserVO 객체 생성

- 1)src/main/java/com.example.vo package 생성
- 2)src/main/java/com.example.vo.UserVO class

```

1861     package com.example.vo;
1862
1863     public class UserVO {
1864         private String name;
1865         private int age;
1866         private String userid;
1867         public String getName() {
1868             return name;
1869         }
1870     }
1871     public void setName(String name) {
1872         this.name = name;
1873     }
1874     public int getAge() {
1875         return age;
1876     }
1877     public void setAge(int age) {
1878         this.age = age;
1879     }
1880     public String getUserid() {
1881         return userid;
1882     }
1883     public void setUserid(String userid) {
1884         this.userid = userid;
1885     }
1886     @Override
1887     public String toString() {
1888         return "UserVO [name=" + name + ", age=" + age + ", userid=" + userid + "];"
1889     }
1890 }
1891
1892

```

9. Validator를 이용한 검증

- 1)Data Command 객체에서 유효성 검사를 할 수 있다.
- 2)UserValidator 객체 생성
- 3)src/main/java/com.example.biz.UserValidator class

```

1898     package com.example.biz;
1899
1900     import org.springframework.validation.Errors;
1901     import org.springframework.validation.Validator;
1902
1903     import com.example.vo.UserVO;
1904
1905     public class UserValidator implements Validator {
1906
1907         @Override
1908         public boolean supports(Class<?> arg0) {

```

```

1909         //검증할 객체의 class 타입 정보를 반환
1910         return UserVO.class.isAssignableFrom(arg0);
1911     }
1912
1913     @Override
1914     public void validate(Object obj, Errors errors) {
1915         System.out.println("검증시작");
1916         UserVO userVO = (UserVO)obj;
1917
1918         String username = userVO.getName();
1919         if(username == null || username.trim().isEmpty()) {
1920             System.out.println("이름의 값이 빠졌습니다.");
1921             errors.rejectValue("name", "No Value");
1922         }
1923
1924         int usage = userVO.getAge();
1925         if(usage == 0) {
1926             System.out.println("나이의 값이 빠졌습니다.");
1927             errors.rejectValue("age", "No Value");
1928         }
1929
1930         String userid = userVO.getUserid();
1931         if(userid == null || userid.trim().isEmpty()) {
1932             System.out.println("아이디의 값이 빠졌습니다.");
1933             errors.rejectValue("userid", "No Value");
1934         }
1935     }
1936 }

```

4)src/main/java/com.example.biz/HomeController.java

```

1939
1940     @RequestMapping(value = "/register", method=RequestMethod.GET)
1941     public String register() {
1942         return "register";
1943     }
1944
1945     @RequestMapping(value = "/register", method=RequestMethod.POST)
1946     public String register(@ModelAttribute("userVO") UserVO userVO, BindingResult result) {
1947         String page = "register_ok";
1948         UserValidator validator = new UserValidator();
1949         validator.validate(userVO, result);
1950         if(result.hasErrors()) {
1951             page = "register";
1952         }
1953         return page;
1954     }
1955

```

5)src/main/webapp/WEB-INF/views/register.jsp

```

1956     <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
1957     <!DOCTYPE html>
1958     <html>
1959     <head>
1960     <meta charset="UTF-8">
1961     <title>회원 가입 폼</title>
1962     </head>
1963     <body>
1964     <form action="/biz/register" method="post">
1965         Name : <input type="text" name="name" /><br />
1966         Age : <input type="number" name="age" /><br />
1967         ID : <input type="text" name="userid" /><br />
1968         <input type="submit" value="가입하기" />
1969     </form>
1970 </body>
1971 </html>
1972

```

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

```

1973
1974     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
1975     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
1976     <c:set var="user" value="${userVO}" />
1977     <!DOCTYPE html">
1978     <html>
1979     <head>
1980     <meta charset="UTF-8">
1981     <title>회원 가입 결과 창</title>

```

```

1983     </head>
1984     <body>
1985         <ul>
1986             <li>이름 : ${user.name}</li>
1987             <li>나이 : ${user.age}</li>
1988             <li>아이디 : ${user.userid}</li>
1989         </ul>
1990     </body>
1991 </html>

```

7)Test

http://localhost:8080/biz/register에서
이름, 나이, 아이디를 모두 입력하면 결과창으로 넘어오고
한 개라도 입력하지 않으면 다시 입력창으로 간다.

10. ValidationUtils class를 이용한 검증

- 1)ValidationUtils class는 validate() method를 좀 더 편리하게 사용할 수 있게 해줌.
- 2)UserValidator.java 수정

```

2003     /*String username = userVO.getName();
2004     if(username == null || username.trim().isEmpty()) {
2005         System.out.println("이름의 값이 빠졌습니다.");
2006         errors.rejectValue("name", "No Value");
2007     }*/
2008
2009     ValidationUtils.rejectIfEmptyOrWhitespace(errors, "name", "No Value");

```

11. @Valid와 @InitBinder 이용하기

- 1)Spring Framework이 대신 검증해 줌
- 2)mvnrepository에서 'hibernate validator'로 검색

```

2016     <dependency>
2017         <groupId>org.hibernate.validator</groupId>
2018         <artifactId>hibernate-validator</artifactId>
2019         <version>6.0.18.Final</version>
2020     </dependency>

```

- 3)pom.xml에 넣고 Maven Clean > Maven Install

- 4)HomeController.java 수정

```

2025     @RequestMapping(value = "/register", method=RequestMethod.POST)
2026     public String register(@ModelAttribute("userVO") @Valid UserVO userVO, BindingResult result) {
2027         String page = "register_ok";
2028         //UserValidator validator = new UserValidator();
2029         //validator.validate(userVO, result);
2030         if(result.hasErrors()) {
2031             page = "register";
2032         }
2033
2034         return page;
2035     }
2036
2037     @InitBinder
2038     protected void initBinder(WebDataBinder binder) {
2039         binder.setValidator(new UserValidator());
2040     }

```

12. Test

http://localhost:8080/biz/register에서
이름, 나이, 아이디를 모두 입력하면 결과창으로 넘어오고
한 개라도 입력하지 않으면 다시 입력창으로 간다.

Task8. Convert J2EE to Spring MVC

1. In J2EE Perspective
2. Project Explorer > right-click > New > Dynamic Web Project
3. Project name : SpringWebDemo > Next
 - Default output folder : build\classes > Next
 - Content directory : WebContent

2057 -Check [Generate web.xml deployment descriptor] > Finish
2058
2059
2060 4. Convert to Maven Project
2061 1)project right-click > Configure > Convert to Maven Project > Finish
2062 2)Project : /SpringWebDemo
2063 3)Group Id : SpringWebDemo
2064 4)Artifact Id : SpringWebDemo
2065 5)version : 0.0.1-SNAPSHOT
2066 6)Packaging : war
2067 7)Finish
2068
2069
2070 5. Add Spring Project Nature
2071 -project right-click > Spring Tools > Add Spring Project Nature
2072
2073
2074 6. 새로 생성된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install
2075 <dependencies>
2076 <dependency>
2077 <groupId>org.springframework</groupId>
2078 <artifactId>spring-context</artifactId>
2079 <version>5.3.12</version>
2080 </dependency>
2081 <dependency>
2082 <groupId>org.junit.jupiter</groupId>
2083 <artifactId>junit-jupiter-api</artifactId>
2084 <version>5.8.1</version>
2085 <scope>test</scope>
2086 </dependency>
2087 <dependency>
2088 <groupId>org.springframework</groupId>
2089 <artifactId>spring-jdbc</artifactId>
2090 <version>5.3.12</version>
2091 </dependency>
2092 </dependencies>
2093 ...
2094 <build>
2095 <plugins>
2096 <plugin>
2097 <artifactId>maven-compiler-plugin</artifactId>
2098 <version>3.8.1</version>
2099 <configuration>
2100 <release>11</release>
2101 </configuration>
2102 </plugin>
2103
2104
2105 7. Spring mvc library 검색 및 설치
2106 1)http://mvnrepository.com에서 'spring mvc'로 검색
2107 2)pom.xml에 추가
2108
2109 <dependency>
2110 <groupId>org.springframework</groupId>
2111 <artifactId>spring-webmvc</artifactId>
2112 <version>5.3.12</version>
2113 </dependency>
2114
2115 3)Java version setting
2116 -SpringWebDemo project > right-click > Properties > Project Facets > Select Java > Change Version 11
2117 -Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close
2118
2119 4)Maven Clean > Maven Install
2120
2121
2122 8. Build path에 config folder 추가
2123 1)project right-click > Build Path > Configure Build Path > Select [Source] tab
2124 2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
2125 3)Folder name : config > Finish > OK > Apply and Close
2126 4)Java Resources > config 폴더 확인
2127
2128
2129 9. config folder에 beans.xml file 생성
2130 1)Spring Perspective로 전환

```

2131 2)config > right-click > New > Other > Spring > Spring Bean Configuration File > beans.xml
2132 3)생성시 beans,context 체크
2133 <?xml version="1.0" encoding="UTF-8"?>
2134 <beans xmlns="http://www.springframework.org/schema/beans"
2135     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2136     xmlns:context="http://www.springframework.org/schema/context"
2137     xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd">
2138
2139
2140
2141 </beans>
2142
2143
2144 10. ContextLoaderListener class 설정
2145 1)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener - ContextLoaderListener]
    를 선택하면 아래의 code가 자동 삽입
2146
2147 <!-- needed for ContextLoaderListener -->
2148 <context-param>
2149     <param-name>contextConfigLocation</param-name>
2150     <param-value>location</param-value>
2151 </context-param>
2152
2153 <!-- Bootstraps the root web application context before servlet initialization -->
2154 <listener>
2155     <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
2156 </listener>
2157
2158 2)아래 code로 변환
2159 <context-param>
2160     <param-name>contextConfigLocation</param-name>
2161     <param-value>classpath:beans.xml</param-value>
2162 </context-param>
2163
2164
2165 11. DispatcherServlet Class 추가
2166 1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherservlet - DispatcherServlet
    declaration] 선택하면 아래의 code가 자동 추가된다.
2167
2168 <!-- The front controller of this Spring Web application, responsible for handling all application requests -->
2169 <servlet>
2170     <servlet-name>springDispatcherServlet</servlet-name>
2171     <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
2172     <init-param>
2173         <param-name>contextConfigLocation</param-name>
2174         <param-value>location</param-value>
2175     </init-param>
2176     <load-on-startup>1</load-on-startup>
2177 </servlet>
2178
2179 <!-- Map all requests to the DispatcherServlet for handling -->
2180 <servlet-mapping>
2181     <servlet-name>springDispatcherServlet</servlet-name>
2182     <url-pattern>url</url-pattern>
2183 </servlet-mapping>
2184
2185 2)아래의 code로 변환
2186 <init-param>
2187     <param-name>contextConfigLocation</param-name>
2188     <param-value>classpath:beans*.xml</param-value>
2189 </init-param>
2190
2191 <servlet-mapping>
2192     <servlet-name>springDispatcherServlet</servlet-name>
2193     <url-pattern>*.do</url-pattern>
2194 </servlet-mapping>
2195
2196
2197 12. mvnrepository에서 'jstl'로 검색 후 설치
2198 1)목록에서 2번째 : 1.2버전
2199
2200 <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->

```

```
2201     <dependency>
2202         <groupId>javax.servlet</groupId>
2203         <artifactId>jstl</artifactId>
2204         <version>1.2</version>
2205     </dependency>
2206
```

2207 2)pom.xml에 붙여넣고 Maven Clean > Maven Install

2208

2209

2210 13. Hello Controller 작성

2211 1)src/com.example.vo package 생성

2212 2)src/com.example.vo.HelloVO class 생성

2213

2214 package com.example.vo;

2215

2216 public class HelloVO {

2217 private String name;

2218

2219 public void setName(String name) {

2220 this.name = name;

2221 }

2222

2223 public String sayHello() {

2224 return "Hello " + name;

2225 }

2226 }

2227

2228 3)src/com.example.controller package 생성

2229 4)com.example.controller.HelloController class 생성

2230

2231 package com.example.controller;

2232

2233 import org.springframework.beans.factory.annotation.Autowired;

2234 import org.springframework.stereotype.Controller;

2235 import org.springframework.ui.Model;

2236 import org.springframework.web.bind.annotation.RequestMapping;

2237

2238 import com.example.vo.HelloVO;

2239

2240 @Controller

2241 public class HelloController {

2242 @Autowired

2243 private HelloVO helloBean;

2244

2245 @RequestMapping("/hello.do")

2246 public String hello(Model model) {

2247 String msg = helloBean.sayHello();

2248 model.addAttribute("greet", msg);

2249 return "hello.jsp";

2250 }

2251 }

2252

2253

2254 14. beans.xml 수정

2255 <context:component-scan base-package="com.example" />

2256

2257 <bean id="helloVO" class="com.example.vo.HelloVO">

2258 <property name="name" value="한지민" />

2259 </bean>

2260

2261

2262 15. WebContent/hello.jsp 생성

2263

2264 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

2265 <!DOCTYPE html>

2266 <html>

2267 <head>

2268 <meta charset="UTF-8">

2269 <title>Insert title here</title>

2270 </head>

2271 <body>

2272 \${greet}

2273 </body>

2274 </html>

```

2275
2276 16. project > right-click > Run As > Run on Server > Finish
2277
2278 17. http://localhost:8080/SpringWebDemo/hello.do
2279     Hello 한지민
2280
2281
2282
2283 -----
2284 Task9. Convert J2EE to Spring MVC
2285 1. In J2EE Perspective
2286 2. Project Explorer > right-click > New > Dynamic Web Project
2287
2288 3. Project name : SpringWebDemo > Next
2289    -Default output folder : build\classes > Next
2290    -Content directory : WebContent
2291    -Check [Generate web.xml deployment descriptor] > Finish
2292
2293
2294 4. Convert to Maven Project
2295    -project right-click > Configure > Convert to Maven Project > Finish
2296
2297
2298 5. Add Spring Project Nature
2299    -project right-click > Spring Tools > Add Spring Project Nature
2300
2301 6. 새로 생성된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install
2302    <dependencies>
2303      <dependency>
2304        <groupId>org.springframework</groupId>
2305        <artifactId>spring-context</artifactId>
2306        <version>5.3.12</version>
2307      </dependency>
2308      <dependency>
2309        <groupId>org.junit.jupiter</groupId>
2310        <artifactId>junit-jupiter-api</artifactId>
2311        <version>5.8.1</version>
2312        <scope>test</scope>
2313      </dependency>
2314      <dependency>
2315        <groupId>org.springframework</groupId>
2316        <artifactId>spring-jdbc</artifactId>
2317        <version>5.3.12</version>
2318      </dependency>
2319      <dependency>
2320        <groupId>javax.servlet</groupId>
2321        <artifactId>jstl</artifactId>
2322        <version>1.2</version>
2323      </dependency>
2324      <dependency>
2325        <groupId>com.oracle</groupId>
2326        <artifactId>ojdbc6</artifactId>
2327        <version>11.2</version>
2328      </dependency>
2329      <dependency>
2330        <groupId>org.springframework</groupId>
2331        <artifactId>spring-webmvc</artifactId>
2332        <version>5.3.12</version>
2333      </dependency>
2334    </dependencies>
2335    ...
2336    ...
2337    <build>
2338      <plugins>
2339        <plugin>
2340          <artifactId>maven-compiler-plugin</artifactId>
2341          <version>3.8.1</version>
2342          <configuration>
2343            <release>11</release>
2344          </configuration>
2345        </plugin>
2346
2347 2)Java version setting
2348    -SpringWebDemo project > right-click > Properties > Project Facets > Select Java > Change Version 11

```



```

2349 -Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close
2350
2351 3)Maven Clean > Maven Install
2352
2353
2354 7. Build path에 config folder 추가
2355 1)project right-click > Build Path > Configure Build Path > Select [Source] tab
2356 2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
2357 3)Folder name : config > Finish > OK > Apply and Close
2358 4)Java Resources > config 폴더 확인
2359
2360
2361 8. config folder에 beans.xml file 생성
2362 1)Spring Perspective로 전환
2363 2)config Folder > right-click > New > Spring Bean Configuration File
2364 3)File name : beans.xml
2365 4)생성시 beans,context, 체크
2366 <?xml version="1.0" encoding="UTF-8"?>
2367 <beans xmlns="http://www.springframework.org/schema/beans"
2368 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2369 xmlns:context="http://www.springframework.org/schema/context"
2370 xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd">
2371
2372
2373
2374 </beans>
2375
2376
2377 9. ContextLoaderListener class 설정
2378 1)web.xml에서 Ctrl + Spacebar를 하면 나타나는 Context Menu에서 [#contextloaderlistener - ContextLoaderListener]
를 선택하면 아래의 코드가 자동 삽입
2379
2380 <!-- needed for ContextLoaderListener -->
2381 <context-param>
2382 <param-name>contextConfigLocation</param-name>
2383 <param-value>location</param-value>
2384 </context-param>
2385
2386 <!-- Bootstraps the root web application context before servlet initialization -->
2387 <listener>
2388 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
2389 </listener>
2390
2391 2)아래 코드로 변환
2392 <context-param>
2393 <param-name>contextConfigLocation</param-name>
2394 <param-value>classpath:beans.xml</param-value>
2395 </context-param>
2396
2397
2398 10. DispatcherServlet Class 추가
2399 1)web.xml에서 Ctrl + Spacebar 하면 나타나는 Context Menu에서 [#dispatcherServlet - DispatcherServlet
declaration] 선택하면 아래의 코드가 자동 추가된다.
2400
2401 <!-- The front controller of this Spring Web application, responsible for handling all application requests -->
2402 <servlet>
2403 <servlet-name>springDispatcherServlet</servlet-name>
2404 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
2405 <init-param>
2406 <param-name>contextConfigLocation</param-name>
2407 <param-value>location</param-value>
2408 </init-param>
2409 <load-on-startup>1</load-on-startup>
2410 </servlet>
2411
2412 <!-- Map all requests to the DispatcherServlet for handling -->
2413 <servlet-mapping>
2414 <servlet-name>springDispatcherServlet</servlet-name>
2415 <url-pattern>url</url-pattern>
2416 </servlet-mapping>
2417
2418 2)아래의 코드로 변환

```

```
2419 <init-param>
2420 <param-name>contextConfigLocation</param-name>
2421 <param-value>classpath:beans*.xml</param-value>
2422 </init-param>
2423
2424 <servlet-mapping>
2425 <servlet-name>springDispatcherServlet</servlet-name>
2426 <url-pattern>*.do</url-pattern>
2427 </servlet-mapping>
2428
2429
```

2430 11. UserVO class 생성

- 2431 1)src/com.example.vo package 생성
- 2432 2)src/com.example.vo.UserVO class 생성

```
2433
2434 package com.example.vo;
2435
2436 public class UserVO {
2437     private String userId;
2438     private String name;
2439     private String gender;
2440     private String city;
2441     public UserVO() {}
2442     public UserVO(String userId, String name, String gender, String city) {
2443         this.userId = userId;
2444         this.name = name;
2445         this.gender = gender;
2446         this.city = city;
2447     }
2448     public String getUserId() {
2449         return userId;
2450     }
2451     public void setUserId(String userId) {
2452         this.userId = userId;
2453     }
2454     public String getName() {
2455         return name;
2456     }
2457     public void setName(String name) {
2458         this.name = name;
2459     }
2460     public String getGender() {
2461         return gender;
2462     }
2463     public void setGender(String gender) {
2464         this.gender = gender;
2465     }
2466     public String getCity() {
2467         return city;
2468     }
2469     public void setCity(String city) {
2470         this.city = city;
2471     }
2472     @Override
2473     public String toString() {
2474         return "UserVO [userId=" + userId + ", name=" + name + ", gender=" + gender + ", city=" + city +
2475             "]\n";
2476     }
2477 }
2478
```

2479 12. UserDao 객체 생성

- 2480 1)src/com.example.dao package 생성
- 2481 2)src/com.example.dao.UserDao interface

```
2482
2483 package com.example.dao;
2484
2485 import java.util.List;
2486
2487 import com.example.vo.UserVO;
2488
2489 public interface UserDao {
2490     void insert(UserVO user);
2491
```

```

2492     List<UserVO> readAll();
2493
2494     void update(UserVO user);
2495
2496     void delete(String id);
2497
2498     UserVO read(String id);
2499 }
2500
2501 -src/com.example.dao.UserDaoImplJDBC.java 생성
2502
2503     package com.example.dao;
2504
2505     import java.sql.ResultSet;
2506     import java.sql.SQLException;
2507     import java.util.List;
2508
2509     import javax.sql.DataSource;
2510
2511     import org.springframework.beans.factory.annotation.Autowired;
2512     import org.springframework.dao.EmptyResultDataAccessException;
2513     import org.springframework.jdbc.core.JdbcTemplate;
2514     import org.springframework.jdbc.core.RowMapper;
2515     import org.springframework.stereotype.Repository;
2516
2517     import com.example.vo.UserVO;
2518
2519     @Repository("userDao")
2520     public class UserDaoImplJDBC implements UserDao {
2521
2522         private JdbcTemplate jdbcTemplate;
2523
2524         @Autowired
2525         public void setDataSource(DataSource dataSource) {
2526             this.jdbcTemplate = new JdbcTemplate(dataSource);
2527         }
2528
2529         class UserMapper implements RowMapper<UserVO> {
2530             public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
2531                 UserVO user = new UserVO();
2532                 user.setUserId(rs.getString("userId"));
2533                 user.setName(rs.getString("name"));
2534                 user.setGender(rs.getString("gender"));
2535                 user.setCity(rs.getString("city"));
2536                 return user;
2537             }
2538         }
2539
2540         @Override
2541         public void insert(UserVO user) {
2542             String SQL = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
2543             jdbcTemplate.update(SQL, user.getUserId(), user.getName(), user.getGender(), user.getCity());
2544
2545             System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" + user.getName());
2546         }
2547
2548         @Override
2549         public List<UserVO> readAll() {
2550             String SQL = "SELECT * FROM users";
2551             List<UserVO> userList = jdbcTemplate.query(SQL, new UserMapper());
2552             return userList;
2553         }
2554
2555         @Override
2556         public void update(UserVO user) {
2557             String SQL = "UPDATE users SET name = ?, gender = ?, city = ? WHERE userid = ?";
2558             jdbcTemplate.update(SQL, user.getName(), user.getGender(), user.getCity(), user.getUserId());
2559             System.out.println("갱신된 Record with ID = " + user.getUserId());
2560         }
2561
2562         @Override
2563         public void delete(String id) {
2564             String SQL = "DELETE FROM users WHERE userid = ?";
2565             jdbcTemplate.update(SQL, id);

```

```

2566         System.out.println("삭제된 Record with ID = " + id);
2567     }
2568
2569     @Override
2570     public UserVO read(String id) {
2571         String SQL = "SELECT * FROM users WHERE userid = ?";
2572         try {
2573             UserVO user = jdbcTemplate.queryForObject(SQL, new Object[] { id }, new UserMapper());
2574             return user;
2575         } catch (EmptyResultDataAccessException e) {
2576             return null;
2577         }
2578     }
2579 }

```

2581 13. UserService 객체 생성

- 2583 1)src/com.example.service package 생성
- 2584 2)src/com.example.service.UserService interface

```

2585
2586     package com.example.service;
2587
2588     import java.util.List;
2589
2590     import com.example.vo.UserVO;
2591
2592     public interface UserService {
2593         void insertUser(UserVO user);
2594
2595         List<UserVO> getUserList();
2596
2597         void deleteUser(String id);
2598
2599         UserVO getUser(String id);
2600
2601         void updateUser(UserVO user);
2602     }

```

2603 3)src/com.example.service.UserServiceImpl.java

```

2604
2605     package com.example.service;
2606
2607     import java.util.List;
2608
2609     import org.springframework.beans.factory.annotation.Autowired;
2610     import org.springframework.stereotype.Service;
2611
2612     import com.example.dao.UserDao;
2613     import com.example.vo.UserVO;
2614
2615     @Service("userService")
2616     public class UserServiceImpl implements UserService {
2617         @Autowired
2618         UserDao userDao;
2619
2620
2621         @Override
2622         public void insertUser(UserVO user) {
2623             this.userDao.insert(user);
2624         }
2625
2626         @Override
2627         public List<UserVO> getUserList() {
2628             return this.userDao.readAll();
2629         }
2630
2631         @Override
2632         public void deleteUser(String id) {
2633             this.userDao.delete(id);
2634         }
2635
2636         @Override
2637         public UserVO getUser(String id) {
2638             return this.userDao.read(id);
2639         }

```

```
2640
2641     @Override
2642     public void updateUser(UserVO user) {
2643         this.userDao.update(user);
2644     }
2645 }
2646
2647
```

14. UserController 객체 생성

1)src/com.example.controller package 생성

2)com.example.controller.UserController class 생성

```
2651
2652     package com.example.controller;
2653
2654     import org.springframework.beans.factory.annotation.Autowired;
2655     import org.springframework.stereotype.Controller;
2656
2657     import com.example.service.UserService;
2658
2659     @Controller
2660     public class UserController {
2661         @Autowired
2662         private UserService userService;
2663
2664         @RequestMapping("/userInfo.do")
2665         public String getUserList(@RequestParam("userId") String userId, Model model) {
2666             UserVO user = userService.getUser(userId);
2667             //System.out.println(user);
2668             model.addAttribute("user", user);
2669             return "userInfo.jsp";
2670         }
2671     }
2672
2673
```

15. config/dbinfo.properties file 생성

```
2674
2675
2676     db.driverClass=oracle.jdbc.driver.OracleDriver
2677     db.url=jdbc:oracle:thin:@localhost:1521:XE
2678     db.username=hr
2679     db.password=hr
2680
2681
```

16. beans.xml 수정

```
2682
2683
2684     <context:component-scan base-package="com.example" />
2685
2686     <context:property-placeholder location="classpath:dbinfo.properties" />
2687     <bean id="dataSource" class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
2688         <property name="driverClass" value="${db.driverClass}" />
2689         <property name="url" value="${db.url}" />
2690         <property name="username" value="${db.username}" />
2691         <property name="password" value="${db.password}" />
2692     </bean>
2693
2694
```

17. WebContent/index.jsp 생성

```
2695
2696
2697     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2698     <c:redirect url="userInfo.do" />
2699
2700
```

18. WebContent/userInfo.jsp 생성

```
2701
2702
2703     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2704     <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2705     <c:set var="user" value="${user}"/>
2706     <!DOCTYPE html>
2707     <html>
2708     <head>
2709     <meta charset="UTF-8">
2710     <title>Insert title here</title>
2711     </head>
2712     <body>
2713     <h1>userInfo.jsp</h1>

```

```
2714     <h2>사용자 정보</h2>
2715     아이디 : ${user.userId} <br />
2716     이름 : ${user.name} <br />
2717     성별 : ${user.gender} <br />
2718     도시 : ${user.city} <br />
2719 </body>
2720 </html>
2721
2722
```

19. project > right-click > Run As > Run on Server > Finish

20. <http://localhost:8080/SpringWebDemo/userinfo.do?userId=scott>

Task10. File Upload with Spring MVC

1. In J2EE Perspective

2. Project Explorer > right-click > New > Dynamic Web Project

3. Project name : FileUploadDemo > Next > Check [Generate web.xml deployment descriptor] > Finish

4. Convert to Maven Project

1)project right-click > Configure > Convert to Maven Project > Finish

5. Add Spring Project Nature

1)project right-click > Spring Tools > Add Spring Project Nature

6. 새로 생성된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install

```
<dependencies>
```

```
  <dependency>
```

```
    <groupId>org.springframework</groupId>
```

```
    <artifactId>spring-context</artifactId>
```

```
    <version>5.3.12</version>
```

```
  </dependency>
```

```
  <dependency>
```

```
    <groupId>org.springframework</groupId>
```

```
    <artifactId>spring-webmvc</artifactId>
```

```
    <version>5.3.12</version>
```

```
  </dependency>
```

```
</dependencies>
```

7. ContextLoaderListener class 설정

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

```
<!-- needed for ContextLoaderListener -->
```

```
<context-param>
```

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

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

```
</context-param>
```

```
<!-- Bootstraps the root web application context before servlet initialization -->
```

```
<listener>
```

```
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
```

```
</listener>
```

2)아래 코드로 변환

```
<context-param>
```

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

```
  <param-value>classpath:applicationContext.xml</param-value>
```

```
</context-param>
```

8. DispatcherServlet Class 추가

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

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

```
<servlet>
```

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

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

```
  <init-param>
```

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

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

```
2786         </init-param>
2787         <load-on-startup>1</load-on-startup>
2788     </servlet>
2789
2790     <!-- Map all requests to the DispatcherServlet for handling -->
2791     <servlet-mapping>
2792         <servlet-name>springDispatcherServlet</servlet-name>
2793         <url-pattern>url</url-pattern>
2794     </servlet-mapping>
2795
```

2)아래의 코드로 변환

```
2797     <init-param>
2798         <param-name>contextConfigLocation</param-name>
2799         <param-value>classpath:beans.xml</param-value>
2800     </init-param>
2801
2802     <servlet-mapping>
2803         <servlet-name>springDispatcherServlet</servlet-name>
2804         <url-pattern>/</url-pattern>
2805     </servlet-mapping>
2806
2807
```

9. FileUpload library 추가

- 1)Apache에서 제공하는 Common FileUpload library를 사용하여 file upload를 처리하기 위한 library
- 2)mvnrepository에서 'common fileupload'라고 검색하여 library 추가

```
2811     <dependency>
2812         <groupId>commons-fileupload</groupId>
2813         <artifactId>commons-fileupload</artifactId>
2814         <version>1.4</version>
2815     </dependency>
2816
```

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

```
2817     <dependency>
2818         <groupId>commons-io</groupId>
2819         <artifactId>commons-io</artifactId>
2820         <version>2.6</version>
2821     </dependency>
2822
```

- 4)Maven Clean > Maven Install

10. Thumbnail Image Library 추가

- 1)mvnrepository에서 'imgscalr-lib'라고 검색하여 library 추가

```
2829     <dependency>
2830         <groupId>org.imgscalr</groupId>
2831         <artifactId>imgscalr-lib</artifactId>
2832         <version>4.2</version>
2833     </dependency>
2834
```

- 2)Maven Clean > Maven Install

11. Build path에 config Foler 추가

- 1)project right-click > Build Path > Configure Build Path > Select [Source] tab
- 2)Click [Add Folder] > Select 현재 project > Click [Create New Folder...]
- 3)Folder name : config > Finish > OK > Apply and Close
- 4)Java Resources > config 폴더 확인

12. config Folder에 applicationContext.xml file 생성

- 1)config > right-click > New > Other > Spring > Spring Bean Configuration File
- 2)Name : applicationContext.xml
- 3)Namespace Tab에서 context, mvc check 할 것

```
2849
2850     <?xml version="1.0" encoding="UTF-8"?>
2851     <beans xmlns="http://www.springframework.org/schema/beans"
2852         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2853         xmlns:context="http://www.springframework.org/schema/context"
2854         xmlns:mvc="http://www.springframework.org/schema/mvc"
2855         xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
```

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

```
<context:component-scan
    base-package="com.example" />
<mvc:annotation-driven />
</beans>
```

13. config Folder에 beans.xml file 생성

- 1)config > right-click > New > Other > Spring > Spring Bean Configuration File
- 2)Name : beans.xml
- 3)beans.xml에 Spring multipartResolver 추가

```
<bean id="multipartResolver"
    class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <property name="maxUploadSize" value="10240000" />
    <property name="defaultEncoding" value="utf-8" />
</bean>
```

14. web.xml에 한글 File Encoding 처리하기

- 1)한글 File이 Upload될 때 File 명이 깨지는 것을 해결하기 위해 web.xml에 아래 내용을 추가한다.

```
<filter>
    <filter-name>encodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
        <param-name>forceEncoding</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

15. View 작성

- 1)WebContent/form.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>file 업로드 예제</h1>
    <form method="post" action="upload" enctype="multipart/form-data">
        <label>email:</label> <input type="text" name="email"> <br>
        <br> <label>file:</label> <input type="file" name="file1">
        <br>
        <br> <input type="submit" value="upload">
    </form>
</body>
</html>
```

16. Service 작성

- 1)src/com.example.service package
- 2)src/com.example.service.FileUploadService.java

```
package com.example.service;

import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Calendar;

import org.springframework.stereotype.Service;
```



```

import org.springframework.web.multipart.MultipartFile;

@Service
public class FileUploadService {
    // 리눅스 기준으로 file 경로를 작성 ( 루트 경로인 /으로 시작한다. )
    // 윈도우라면 workspace의 드라이브를 파악하여 JVM이 알아서 처리해준다.
    // 따라서 workspace가 C드라이브에 있다면 C드라이브에 upload 폴더를 생성해 놓아야 한다.
    private static final String SAVE_PATH = "/upload";
    private static final String PREFIX_URL = "/upload/";

    public String restore(MultipartFile multipartFile) {
        String uri = null;

        try {
            // file 정보
            String originFilename = multipartFile.getOriginalFilename();
            String extName = originFilename.substring(originFilename.lastIndexOf("."), originFilename.length());
            Long size = multipartFile.getSize();

            // 서버에서 저장 할 file 이름
            String saveFileName = genSaveFileName(extName);

            System.out.println("originFilename : " + originFilename);
            System.out.println("extensionName : " + extName);
            System.out.println("size : " + size);
            System.out.println("saveFileName : " + saveFileName);

            writeFile(multipartFile, saveFileName);
            uri = PREFIX_URL + saveFileName;
        } catch (IOException e) {
            // 원래라면 RuntimeException 을 상속받은 예외가 처리되어야 하지만
            // 편의상 RuntimeException을 던진다.
            // throw new FileUploadException();
            throw new RuntimeException(e);
        }
        return uri;
    }

    // 현재 시간을 기준으로 file 이름 생성
    private String genSaveFileName(String extName) {
        String fileName = "";

        Calendar calendar = Calendar.getInstance();
        fileName += calendar.get(Calendar.YEAR);
        fileName += calendar.get(Calendar.MONTH);
        fileName += calendar.get(Calendar.DATE);
        fileName += calendar.get(Calendar.HOUR);
        fileName += calendar.get(Calendar.MINUTE);
        fileName += calendar.get(Calendar.SECOND);
        fileName += calendar.get(Calendar.MILLISECOND);
        fileName += extName;

        return fileName;
    }

    // file을 실제로 write 하는 메서드
    private boolean writeFile(MultipartFile multipartFile, String saveFileName)
        throws IOException{
        boolean result = false;

        byte[] data = multipartFile.getBytes();
        FileOutputStream fos = new FileOutputStream(SAVE_PATH + "/" + saveFileName);
        fos.write(data);
        fos.close();

        return result;
    }
}

```

3)SAVE_PATH는 file을 저장할 위치를 가리킨다.

-일반적으로 server는 Linux 기반이므로 Linux 경로명을 사용하는 것이 좋다.

3004 -즉 file을 root 경로인 / 아래의 upload folder에 저장하겠다는 의미인데, Windows에서는 JVM이 알아서 workspace가 존재하는
drive의 위치를 찾아서 drive를 root 경로로 하여 upload folder에 저장한다.
3005 -예를들어 Eclipse workspace가 C drive에 있다면 C drive의 upload folder에 file이 저장될 것이다.
3006 4)PREFIX_URL은 저장된 file을 JSP에서 불러오기 위한 경로를 의미한다.
3007 5)MultipartFile 객체는 file의 정보를 담고 있다.
3008 6)uri을 반환하는 이유는 view page에서 바로 image file을 보기 위함이다.
3009 -만약 DB에서 image 경로를 저장 해야 한다면, 이와 같이 uri을 반환하면 좋을 것이다.
3010 7)현재 시간을 기준으로 file 이름을 바꾼다.
3011 -이렇게 하는 이유는, 여러 사용자가 올린 file의 이름이 같을 경우 덮어 씌어지는 문제가 발생하기 때문이다.
3012 -따라서 file 이름이 중복될 수 있는 문제를 해결하기 위해 ms단위의 시스템 시간을 이용하여 file 이름을 변경한다.
3013 8)FileOutputStream 객체를 이용하여 file을 저장한다.

17. Controller 작성

- 1)com.example.controller package
- 2)com.example.controller.FileUploadController.java

```
package com.example.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

import com.example.service.FileUploadService;

@Controller
public class FileUploadController {
    @Autowired
    FileUploadService fileUploadService;

    @RequestMapping("/form")
    public String form() {
        return "form.jsp";
    }

    @RequestMapping(value = "/upload", method = RequestMethod.POST)
    public String upload(@RequestParam("email") String email, @RequestParam("file1") MultipartFile file,
        Model model) {
        String uri = fileUploadService.restore(file);
        model.addAttribute("uri", uri);
        return "result.jsp";
    }
}
```

18. WebContent/result.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>Upload completed</h1>
<div class="result-images">

</div>
<p>
<a href='/FileUploadDemo/form'> 다시 업로드 하기 </a>
</p>
</body>
</html>
```

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

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

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

```

3076
3077 21. 문제점 및 해결
3078 1)Upload Folder(C:/upload)를 보면 File이 Upload된 것을 확인할 수 있지만, 결과 화면을 보면 Image가 제대로 출력 되지 않을 것이다.
3079 2)Image File을 right-click하여 경로를 보면 아마 다음과 같을 것이다.
3080 3)http://localhost:8080/FileUploadDemo/upload/업로드한 파일
3081 4)File을 저장할 때 [upload]라는 Folder에 저장을 했는데, File을 저장할 때의 Upload는 C Drive 내의 [upload] Folder이고,
3082 5)위 URL에서 [upload]는 Application 상 경로에 있는 upload이므로 WEB-INF 폴더의 하위 folder로서의 upload를 의미한다.
3083 6)즉 실제 File이 저장된 Server 상의 위치( 물리 주소 )와 Application에서 보여주고자 하는 File 경로( 가상 주소 )가 일치하지 않은 것이다.
3084 7)따라서 실제 File이 저장되어 있는 위치와 Application 상의 위치를 일치시키는 작업이 필요하다.
3085 8)beans.xml에 물리 주소와 가상 주소를 mapping 해주는 code를 추가하도록 해야한다.
3086
3087     <!-- resource mapping -->
3088     <!-- location : 물리적 주소 / mapping : 가상 주소 -->
3089     <mvc:resources location="file:///C:/upload/" mapping="/upload/*"/>
3090
3091 9)이제 정상적으로 result.jsp에서 image가 출력될 것이다.
3092
3093
3094 22. Multiple File Upload
3095 1)이번에는 여러 개의 File을 Upload 할 수 있는 Multiple Upload를 알아보자.
3096 2)수정할 부분은 <input> tag와 Controller에서 MultipartFile 객체를 받는 Parameter 부분 두 곳인데, 필요한 부분만 보자.
3097 3)form.jsp
3098
3099     <input type="file" name="files" multiple>
3100
3101 4)<input> 태그에서는 multiple 속성만 추가하면 된다.
3102 5)"File선택"을 클릭하면 ctrl 키를 눌러서 여러 개의 File을 선택할 수 있다.
3103
3104 6)FileUploadController
3105
3106     @RequestMapping( "/upload" )
3107     public String upload(@RequestParam String email,
3108         @RequestParam(required=false) List<MultipartFile> files, Model model) {
3109
3110         ...
3111
3112     }
3113
3114 7)Controller에서는 여러 개의 File을 받기 때문에 MultipartFile을 List로 받아야 한다.

```