HOL : Spring MVC

Task1. Spring MVC Demo
1. Package Explorer > right-click > New > Other > Spring > Spring Legacy Project
2. Select Spring MVC Project
3. Project name : HelloWorldWeb > Next
4. Enter a topLevelPackage : com.example.biz > Finish
5. pom.xml 수정하기

```
<properties>
    <java-version>1.8</java-version>
    <org.springframework-version>5.2.0.RELEASE</org.springframework-version>
    <org.aspectj-version>1.9.4</org.aspectj-version>
    <org.slf4j-version>1.7.28</org.slf4j-version>
</properties>
    ...
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>javax.servlet.jsp-api</artifactId>
    <version>2.3.3</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

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

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

8. HelloWorldWeb Project right-click > Run As > Run on Server > Finish

9. http://localhost:8080/biz/

   Hello world!

   The time on the server is 2019년 6월 11일 (화) 오후 11시 40분 58초.<--원래 한글 깨짐

10. 한글 깨짐을 수정하는 것은 src/main/webapp/WEB-INF/views/home.jsp에서
    <%@ page session="false" pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>로 수정

11. Context name 변경하기

1)Package Explorer에서 Servers/Tomcat v9.0 Server at localhost-config/server.xml에서 다음과 같이 수정한다.
-path="/biz" --> path="/demo"
<Context docBase="HelloWorldWeb" path="/demo" reloadable="true" source="org.eclipse.jst.jee.server:HelloWorldWeb"/>

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

Task2. resources Folder 이용하기

1. Image 경로 알아내기
1)src/main/webapp/resources/에 images Folder를 STS Package Explorer에서 생성한다.
2)Download받은 image를 src/main/webapp/resources/images/에 넣는다.
3)home.jsp에 아래 code를 추가한다.

4)Image가 잘 나온다.

2. Image 경로 변경
1)apple.jpg image 경로를 src/main/webapp/images/로 이동.
2)하지만 이렇게 하면 image가 보이지 않는다.
3)왜냐하면, servlet-context.xml에서 resource의 경로는 <resources mapping="/resources/**" location="/resources/" />이기 때문.
4)즉, 기본적으로 resources folder 아래에서 resource를 찾는다.

3. <resources />추가
1)다시 resources Folder 하위로 images Folder를 복사
2)/src/main/webapp/하위에 images Folder를 생성하고 image를 넣고 home.jsp에 아래의 code를 추가한다.

3)하지만 아래의 image는 보이지 않는다.
4)왜냐하면 새로 추가한 images Folder는 servlet-context.xml에서 설정하지 않았기 때문.
5)Image를 보이게 하기 위해 servlet-context.xml에 아래의 Code를 추가한다.
<resources mapping="/resources/**" location="/resources/" />
<resources mapping="/images/**" location="/images/" />
6)src/main/webapp/images Folder 추가
7)Project right-click > Run As > Run on Server > Restart >
-Image가 제대로 2개가 나온다.

Task3. Controller Class 제작하기

1. 제작순서
1)@Controller를 이용한 class 생성
2)@RequestMapping을 이용한 요청 경로 지정
3)요청 처리 method 구현
4)View 이름 return

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

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

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

```
205             <meta charset="UTF-8">
206             <title>Insert title here</title>
207          </head>
208          <body>
209             아이디 : ${userid} <br />
210             패스워드 : ${passwd}
211          </body>
212       </html>
213
214    3)http://localhost:8080/demo/demo  --> /demo.jsp
215       아이디 : example
216       패스워드 : 12345678
217
218
219 5. Controller class에 @RequestMapping 적용
220    1)src/main/java/com.example.biz.StudentController.java 생성
221
222       package com.example.biz;
223
224       import org.springframework.stereotype.Controller;
225       import org.springframework.web.bind.annotation.RequestMapping;
226       import org.springframework.web.bind.annotation.RequestMethod;
227       import org.springframework.web.servlet.ModelAndView;
228
229       @Controller
230       @RequestMapping("/bbs")
231       public class StudentController {
232
233          @RequestMapping(value="/get", method = RequestMethod.GET)
234          public ModelAndView getStudent() {
235
236             ModelAndView mav = new ModelAndView();
237             mav.setViewName("/bbs/get");   //  /WEB-INF/views/bbs/get.jsp
238             mav.addObject("name", "한지민");
239             mav.addObject("age", 25);
240             return mav;
241          }
242       }
243
244    2)src/main/webapp/WEB-INF/views/bbs/get.jsp
245       <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
246       <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
          "http://www.w3.org/TR/html4/loose.dtd">
247       <html>
248       <head>
249       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
250       <title>Insert title here</title>
251       </head>
252       <body>
253          학생 이름 : ${name} <br />
254          학생 나이 : ${age}
255       </body>
```

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

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

```
360     public String confirm(@RequestParam("userid") String userid,
361                           @RequestParam("passwd") String passwd,
362                           @RequestParam("name") String name,
363                           @RequestParam("age") int age,
364                           @RequestParam("gender") String gender ,Model model) {
365
366        model.addAttribute("userid", userid);
367        model.addAttribute("passwd", passwd);
368        model.addAttribute("name", name);
369        model.addAttribute("age", age);
370        model.addAttribute("gender", gender);
371        return "confirm";   //  /WEB-INF/views/confirm.jsp
372     }
373
374  2)src/main/webapp/WEB-INF/views/confirm.jsp
375
376     <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
377     <!DOCTYPE html>
378     <html>
379        <head>
380           <meta charset="UTF-8">
381           <title>Insert title here</title>
382        </head>
383        <body>
384           아이디 : ${userid} <br />
385           패스워드 : ${passwd} <br />
386           사용자 이름 : ${name} <br />
387           나이 : ${age} <br />
388           성별 : ${gender} <br />
389        </body>
390     </html>
391
392  3)localhost:8080/biz/confirm?name=한지민&gender=여성&age=25&userid=jimin&passwd=1234
393     아이디 : jimin
394     패스워드 : 1234
395     사용자 이름 : 한지민
396     나이 : 25
397     성별 : 여성
398
399
400 11. Data Commander 객체 이용하기1
401  1)src/main/java/com.example.vo.UserVO.java 생성
402
403     package com.example.vo;
404
405     public class UserVO {
406        private String userid;
407        private String passwd;
408        private String name;
409        private int age;
410        private String gender;
411
```

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

```
            @RequestParam("age") int age,
            @RequestParam("gender") String gender  ,Model model) {

        UserVO userVO = new UserVO();
        userVO.setUserid(userid);
        userVO.setPasswd(passwd);
        userVO.setName(name);
        userVO.setAge(age);
        userVO.setGender(gender);

        model.addAttribute("userVO", userVO);

        return "confirm1";    //  /WEB-INF/views/confirm1.jsp
    }
```

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

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="user" value="${userVO}"/>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>confirm1.jsp</h1>
    <h2>사용자 정보</h2>
    아이디 : ${user.userid} <br />
    패스워드 : ${user.passwd} <br />
    이름 : ${user.name} <br />
    나이 : ${user.age} <br />
    성별 : ${user.gender}
</body>
</html>
```

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

confirm1.jsp

사용자 정보

아이디 : jimin
패스워드 : 1234
사용자 이름 : 한지민
나이 : 25
성별 : 여성

12. Data Commander 객체 이용하기2

1)RequestController.java

```
    @RequestMapping(value="/confirm", method=RequestMethod.GET)
    public String confirm(UserVO userVO) {

       return "confirm2";   //  /WEB-INF/views/confirm2.jsp
    }
```

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

```
    <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
    <c:set var="user" value="${userVO}"/>
    <!DOCTYPE html>
    <html>
    <head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
    </head>
    <body>
       <h1>confirm2.jsp</h1>
       <h2>사용자 정보</h2>
       아이디 : ${user.userid} <br />
       패스워드 : ${user.passwd} <br />
       이름 : ${user.name} <br />
       나이 : ${user.age} <br />
       성별 : ${user.gender}
    </body>
    </html>
```

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

confirm2.jsp

사용자 정보

아이디 : jimin
패스워드 : 1234
사용자 이름 : 한지민
나이 : 25
성별 : 여성

13. @PathVariable 이용하기

1)RequestController.java

```
    @RequestMapping(value="/confirm/{userid}/{passwd}/{name}/{age}/{gender}",
    method=RequestMethod.GET)
    public String confirm(@PathVariable String userid, @PathVariable String passwd,
                          @PathVariable String name, @PathVariable int age,
                    @PathVariable String gender,Model model) {
       model.addAttribute("userInfo", new UserVO(userid, passwd, name, age, gender));
       return "confirm3";
    }
```

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

      <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
      <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
      <c:set var="user" value="${userInfo}"/>
      <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
      "http://www.w3.org/TR/html4/loose.dtd">
      <html>
      <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Insert title here</title>
      </head>
      <body>
         <h1>confirm3.jsp</h1>
         <h2>사용자 정보</h2>
         아이디 : ${user.userid} <br />
         패스워드 : ${user.passwd} <br />
         이름 : ${user.name} <br />
         나이 : ${user.age} <br />
         성별 : ${user.gender}
      </body>
      </html>

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

      사용자 정보

      아이디 : jimin
      패스워드 : 1234
      사용자 이름 : 한지민
      나이 : 25
      성별 : 여성


------------------------------
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 />
```

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

```
        <h1>사용자 정보</h1>
        <ul>
          <li>아이디 : ${user.userid}</li>
          <li>패스워드 : ${user.passwd}</li>
          <li>이름 : ${user.name}</li>
          <li>나이 : ${user.age}</li>
          <li>성별 : ${user.gender}</li>
        </ul>
      </body>
      </html>

  5)Spring에서 POST 방식으로 Data를 보낼 때 한글깨짐 현상 발생
  6)해결방법
  7)web.xml

      <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>
      </filter>
      <filter-mapping>
        <filter-name>encodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
      </filter-mapping>

  8)http://localhost:8080/biz/resources/register.html -->
  9)http://localhost:8080/biz/register
      사용자 정보

      아이디 : jimin
      패스워드 : 1234
      사용자 이름 : 한지민
      나이 : 25
      성별 : 여성


3. redirect: 키워드 이용하기
  1)src/main/java/com.example.biz/HomeController.java

      @RequestMapping("/verify")
      public String verify(HttpServletRequest request, Model model) {
        String userid = request.getParameter("userid");
        if(userid.equals("admin")) {      //만일 userid가 admin 이면 /admin으로 리다이렉트
          return "redirect:admin";
        }
        return "redirect:user";           //만일 userid가 admin 이 아니면 /user로 리다이렉트
        //return "redirect:http://www.naver.com";    //절대 경로도 가능
      }

```

```
    @RequestMapping("/admin")
    public String verify1(Model model) {
       model.addAttribute("authority", "관리자권한");
       return "admin";
    }

    @RequestMapping("/user")
    public String verify2(Model model) {
       model.addAttribute("authority", "일반사용자");
       return "user";
    }
```

2)/src/main/webapp/WEB-INF/views/admin.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>관리자 페이지</h1>
       권한 : ${authority}
    </body>
    </html>
```

3)/src/main/webapp/WEB-INF/views/user.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>일반 사용자 페이지</h1>
       권한 : ${authority}
    </body>
    </html>
```

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

---

Task6. Database와 연동하기
1. Package Explorer > right-click > New > Spring Legacy Project
2. Select Spring MVC Project
3. Project name : MVCDemo1 > Next
4. Enter a topLevelPackage : com.example.biz > Finish
5. pom.xml 수정하기

```
    <properties>
      <java-version>1.8</java-version>
      <org.springframework-version>5.2.0.RELEASE</org.springframework-version>
      <org.aspectj-version>1.9.4</org.aspectj-version>
      <org.slf4j-version>1.7.28</org.slf4j-version>
    </properties>
      ...
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>4.0.1</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet.jsp</groupId>
      <artifactId>javax.servlet.jsp-api</artifactId>
      <version>2.3.3</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
    </dependency>


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

7. MVCDemo1 Project > right-click > Properties > Project Facets > Select Java > Change Version 1.8
   Select Runtimes Tab > Check Apache Tomcat v9.0 > Click Apply and Close


8. Create Table in MariaDB
      CREATE TABLE Member
      (
         userid          VARCHAR(20),
         username          VARCHAR(20)    NOT NULL,
         userage          TINYINT    NOT NULL,
         gender        VARCHAR(10)    NOT NULL,
         city             VARCHAR(50),
         CONSTRAINT member_userid_pk   PRIMARY KEY(userid)
      );
      -반드시 [test] Database의 조합을 utf8_general_ci로 맞출 것
      -반드시 Member Table의 기본조합이 utf8_general_ci 임을 확인할 것


9. src/main/webapp/static folder 생성
   1)src/main/webapp/static/css folder
   2)src/main/webapp/static/images folder
   3)src/main/webapp/static/js folder
```

```
    -jquery-1.12.4.js
  4)src/main/webapp/static/register.html
    <!DOCTYPE html>
    <html lang="en">
    <head>
      <meta charset="UTF-8">
      <title>회원 가입</title>
    </head>
    <body>
      <h1>회원 가입 창</h1>
      <form action="/biz/create" method="post">
        <ul>
          <li>ID : <input type="text" name="userid" /></li>
          <li>이름 : <input type="text" name="username" /></li>
          <li>나이 : <input type="number" name="age" /></li>
          <li>성별 : <input type="radio" name="gender" value="남성"/>남성
                    <input type="radio" name="gender" value="여성"/>여성</li>
          <li>거주지 : <input type="text" name="city" /></li>
          <li><input type="submit" value="가입하기" /></li>
        </ul>
      </form>
    </body>
    </html>


10. src/main/webapp/WEB-INF/spring/appServlet/sevlet-context.xml 수정
  <resources mapping="/static/**" location="/static/" /> 추가

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


11. src/main/resources/mariadb.properties
  db.driverClass=org.mariadb.jdbc.Driver
  db.url=jdbc:mariadb://localhost:3306/test
  db.username=root
  db.password=javamariadb


12. Spring JDBC 설치
  1)JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>5.2.0.RELEASE</version>
    </dependency>

  2)pom.xml에 붙여 넣고 Maven Install 하기
    [INFO] BUILD SUCCESS
```

13. MariaDB Jdbc Driver library 검색 및 설치
1)Maven Repository 에서 'mariadb'로 검색하여 MariaDB Java Client를 설치한다.

```
<dependency>
    <groupId>org.mariadb.jdbc</groupId>
    <artifactId>mariadb-java-client</artifactId>
    <version>2.5.1</version>
</dependency>
```

2)pom.xml에 붙여 넣고 Maven Install 하기
[INFO] BUILD SUCCESS

14. src/main/webapp/WEB-INF/spring/root-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  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">

  <!-- Root Context: defines shared resources visible to all other web components -->
  <context:property-placeholder location="classpath:mariadb.properties"/>
  <bean id="dataSource" class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
    <property name="driverClass" value="${db.driverClass}" />
    <property name="url" value="${db.url}" />
    <property name="username" value="${db.username}" />
    <property name="password" value="${db.password}" />
  </bean>

  <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="dataSource" />
  </bean>
</beans>
```

15. src/test/java/com.example.biz/TestApp class 생성
1)com.example.biz > right-click > New > JUnit Test Case
2)Select [New JUnit 4 test]
3)Name : TestApp
4)Finish

```
package com.example.biz;

import org.junit.Before;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;
import org.springframework.jdbc.core.JdbcTemplate;

public class TestApp {
```

```
978         private ApplicationContext ctx;
979
980         @Before
981         public void init() {
982            this.ctx = new
               GenericXmlApplicationContext("file:src/main/webapp/WEB-INF/spring**/root-context.xml");
983         }
984         @Test
985         public void test() {
986            JdbcTemplate jdbcTemplate = this.ctx.getBean("jdbcTemplate", JdbcTemplate.class);
987            System.out.println(jdbcTemplate);
988         }
989      }
990
991   5)Run as > JUnit Test > Green bar
992
993
994 16. package 생성
995   1)src/main/java/com.example.vo
996   2)src/main/java/com.example.dao
997   3)src/main/java/com.example.service
998
999
1000 17. src/com.example.vo.MemberVO class 생성
1001
1002   package com.example.vo;
1003
1004   public class MemberVO {
1005      private String userid;
1006      private String username;
1007      private int age;
1008      private String gender;
1009      private String city;
1010
1011      public MemberVO() {}
1012
1013      public MemberVO(String userid, String username, int age, String gender, String city) {
1014         this.userid = userid;
1015         this.username = username;
1016         this.age = age;
1017         this.gender = gender;
1018         this.city = city;
1019      }
1020
1021      public String getUserid() {
1022         return userid;
1023      }
1024
1025      public void setUserid(String userid) {
1026         this.userid = userid;
1027      }
1028
```

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

```
        int update(MemberVO memberVo);
        int delete(String userid);
    }

  2)MemberDaoImpl.java
    package com.example.dao;

    import java.util.List;

    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.jdbc.core.JdbcTemplate;
    import org.springframework.stereotype.Repository;

    import com.example.vo.MemberVO;

    @Repository("memberDao")
    public class MemberDaoImpl implements MemberDao {
        @Autowired
        JdbcTemplate jdbcTemplate;

    @Override
    public int create(MemberVO memberVo) {
        return 0;
    }

    @Override
    public MemberVO read(String userid) {
        return null;
    }

    @Override
    public List<MemberVO> readAll() {
        return null;
    }

    @Override
    public int update(MemberVO memberVo) {
        return 0;
    }

    @Override
    public int delete(String userid) {
        return 0;
    }
  }

19. com.example.service
  1)MemberService interface
    package com.example.service;

    import java.util.List;

```

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

20. com.example.biz
1)HomeController.java

```
package com.example.biz;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;

import com.example.service.MemberService;

/**
 * Handles requests for the application home page.
 */
@Controller
public class HomeController {
   @Autowired
   MemberService memberService;
}
```

21. Data Insert
1)/src/main/webapp/static/register.html
2)com.example.biz/HomeController.java

```
@Controller
public class HomeController {
   @Autowired
   MemberService memberService;

   @RequestMapping(value = "/create", method = RequestMethod.POST)
   public String home(MemberVO memberVo, Model model) {
      int row = this.memberService.create(memberVo);
      if(row == 1) model.addAttribute("status", "Insert Success");
      else model.addAttribute("status", "Insert Failure");
      return "create";    //   /WEB-INF/views/create.jsp
   }
}
```

3)com.example.service/MemberServiceImpl.java

```
@Service("memberService")
public class MemberServiceImpl implements MemberService {
   @Autowired
   MemberDao memberDao;

   @Override
   public int create(MemberVO memberVo) {
      return this.memberDao.create(memberVo);
   }
}
```

4)com.example.dao.MemberDaoImpl.java

```
@Repository("memberDao")
public class MemberDaoImpl implements MemberDao {
   @Autowired
   JdbcTemplate jdbcTemplate;

   @Override
   public int create(MemberVO memberVo) {
      String sql = "INSERT INTO Member VALUES(?,?,?,?,?)";
      return this.jdbcTemplate.update(sql, memberVo.getUserid(),
            memberVo.getUsername(), memberVo.getAge(),
            memberVo.getGender(), memberVo.getCity());
   }
}
```

5)views/create.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>${status}</h1>
</body>
</html>
```

22. POST 발송시 한글 깨짐 처리하기

1)web.xml

```
<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>
</filter>
<filter-mapping>
   <filter-name>encodingFilter</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

23. Test

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

2)http://localhost:8080/biz/create

Insert Success

24. Data Select

1)HomeController.java

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

2)MemberServiceImpl.java

```
@Override
public MemberVO read(String userid) {
   return this.memberDao.read(userid);
}
```

3)MemberDaoImpl.java

```
@Override
public MemberVO read(String userid) {
   String sql = "SELECT * FROM Member WHERE userid = ?";
   return this.jdbcTemplate.queryForObject(sql, new Object[] {userid},
                  new MyRowMapper());
}

class MyRowMapper implements RowMapper<MemberVO>{
   @Override
   public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
      MemberVO memberVo = new MemberVO(rs.getString("userid"),
            rs.getString("username"), rs.getInt("userage"),
            rs.getString("gender"), rs.getString("city"));
      return memberVo;
   }
}
```

4)views/view.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<c:set var="user" value="${member}" />
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>Insert title here</title>
   <script src="/biz/static/js/jquery-1.12.4.js"></script>
   <script>
      $(function(){
         $("#btnList").bind("click", function(){
            location.href = "/biz/list";
         });
```

```
            $("#btnDelete").bind("click", function(){
               location.href = "/biz/delete/${user.userid}";
            });
         });
      </script>
   </head>
   <body>
      <h1>${user.username}의정보</h1>
      <form action="/biz/update" method="post">
         <input type="hidden" name="userid" value = "${user.userid}" />
         <ul>
            <li>아이디 : ${user.userid }</li>
            <li>나이 : <input type='number' name="age" value='${user.age}' /></li>
            <li>성별 : <c:if test='${user.gender eq "남성"}'>
                  <input type="radio" name="gender" value="남성" checked />남성  
                  <input type="radio" name="gender" value="여성" />여성
                  </c:if>
                  <c:if test='${user.gender eq "여성"}'>
                  <input type="radio" name="gender" value="남성" />남성  
               <input type="radio" name="gender" value="여성" checked />여성
            </c:if>
            </li>
            <li>거주지 : <input type="text" name="city" value="${user.city }" /></li>
            <li><input type='submit' value='수정하기' /></li>
            <li><input type='button' value='삭제하기'  id="btnDelete"/></li>
            <li><input type='button' value='목록으로'  id="btnList"/></li>
         </ul>
      </form>
   </body>
   </html>
```

5)Test

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

25. Data List

1)HomeController.java

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

2)MemberServiceImpl.java

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

3)MemberDaoImpl.java

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

4)iews/list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!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="${userlist}" var="user">
            <tr>
               <td><a
               href="/biz/view/${user.userid}">${user.userid}</a></td><td>${user.username}</td>
               <td>${user.age}</td><td>${user.gender }</td>
               <td>${user.city }</td>
            </tr>
         </c:forEach>
      </tbody>
   </table>
</body>
</html>
```

5)Test

http://localhost:8080/biz/list

26. Data Delete

1)HomeController.java

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

2)MemberServiceImpl.java

```
@Override
public int delete(String userid) {
   return this.memberDao.delete(userid);
}
```

3)MemberDaoImpl.java

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

4)Test
http://localhost:8080/biz/delete/chulsu

27. Data Update
1)HomeController.java

```
@RequestMapping(value = "/update", method = RequestMethod.POST)
public String update(@RequestParam("userid") String userid,
      @RequestParam("age") int age,
      @RequestParam("gender") String gender,
      @RequestParam("city") String city) {
   this.memberService.update(
         new MemberVO(userid, "", age, gender, city));
   return "redirect:/list";
}
```

2)MemberServiceImpl.java

```
@Override
public int update(MemberVO memberVo) {
   return this.memberDao.update(memberVo);
}
```

3)MemberDaoImpl.java

```
@Override
public int update(MemberVO memberVo) {
   String sql = "UPDATE Member SET userage = ?, gender = ?, city = ? " +
                "WHERE userid = ?";
   return this.jdbcTemplate.update(sql, memberVo.getAge(),
         memberVo.getGender(), memberVo.getCity(), memberVo.getUserid());
}
```

4)Test
http://localhost:8080/biz/list에서

해당 ID Click
데이터 수정
[수정하기] button click

28. All Codes
1)HomeController.java

```
package com.example.biz;

import java.util.List;

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

import com.example.service.MemberService;
import com.example.vo.MemberVO;

/**
 * Handles requests for the application home page.
 */
@Controller
public class HomeController {
   @Autowired
   MemberService memberService;

   @RequestMapping(value = "/create", method = RequestMethod.POST)
   public String home(MemberVO memberVo, Model model) {
      int row = this.memberService.create(memberVo);
      if(row == 1) model.addAttribute("status", "Insert Success");
      else model.addAttribute("status", "Insert Failure");
      return "create";    //   /WEB-INF/views/create.jsp
   }

   @RequestMapping(value = "/view/{userid}", method = RequestMethod.GET)
   public String view(@PathVariable String userid, Model model) {
      MemberVO memberVo = this.memberService.read(userid);
      model.addAttribute("member", memberVo);
      return "view";
   }

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

```
1547
1548        @RequestMapping(value = "/delete/{userid}", method = RequestMethod.GET)
1549        public String delete(@PathVariable String userid) {
1550           this.memberService.delete(userid);
1551           return "redirect:/list";
1552        }
1553
1554        @RequestMapping(value = "/update", method = RequestMethod.POST)
1555        public String update(@RequestParam("userid") String userid,
1556              @RequestParam("age") int age,
1557              @RequestParam("gender") String gender,
1558              @RequestParam("city") String city) {
1559           this.memberService.update(
1560                 new MemberVO(userid, "", age, gender, city));
1561           return "redirect:/list";
1562        }
1563     }
1564
1565  2)MemberServiceImpl.java
1566
1567     package com.example.service;
1568
1569     import java.util.List;
1570
1571     import org.springframework.beans.factory.annotation.Autowired;
1572     import org.springframework.stereotype.Service;
1573
1574     import com.example.dao.MemberDao;
1575     import com.example.vo.MemberVO;
1576
1577     @Service("memberService")
1578     public class MemberServiceImpl implements MemberService {
1579        @Autowired
1580        MemberDao memberDao;
1581
1582        @Override
1583        public int create(MemberVO memberVo) {
1584           return this.memberDao.create(memberVo);
1585        }
1586
1587        @Override
1588        public MemberVO read(String userid) {
1589           return this.memberDao.read(userid);
1590        }
1591
1592        @Override
1593        public List<MemberVO> readAll() {
1594           return this.memberDao.readAll();
1595        }
1596
1597        @Override
1598        public int update(MemberVO memberVo) {
```

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

```
1649
1650          @Override
1651          public List<MemberVO> readAll() {
1652             String sql = "SELECT * FROM Member ORDER BY userid DESC";
1653             return this.jdbcTemplate.query(sql, new MyRowMapper());
1654          }
1655
1656          @Override
1657          public int update(MemberVO memberVo) {
1658             String sql = "UPDATE Member SET userage = ?, gender = ?, city = ? " + "WHERE userid = ?";
1659             return this.jdbcTemplate.update(sql, memberVo.getAge(), memberVo.getGender(),
                 memberVo.getCity(),
1660                   memberVo.getUserid());
1661          }
1662
1663          @Override
1664          public int delete(String userid) {
1665             String sql = "DELETE FROM Member WHERE userid = ?";
1666             return this.jdbcTemplate.update(sql, userid);
1667          }
1668       }
1669
1670
1671 ------------------------------------------------------------
1672 Task7. Form Data Validation
1673 1. Package Explorer > right-click > New > Other > Spring > Spring Legacy Project
1674 2. Select Spring MVC Project
1675 3. Project name : FormValidationDemo > Next
1676 4. Enter a topLevelPackage : com.example.biz > Finish
1677 5. pom.xml 수정하기
1678    <properties>
1679       <java-version>1.8</java-version>
1680       <org.springframework-version>5.2.0.RELEASE</org.springframework-version>
1681       <org.aspectj-version>1.9.4</org.aspectj-version>
1682       <org.slf4j-version>1.7.28</org.slf4j-version>
1683    </properties>
1684       ...
1685    <dependency>
1686       <groupId>javax.servlet</groupId>
1687       <artifactId>javax.servlet-api</artifactId>
1688       <version>4.0.1</version>
1689       <scope>provided</scope>
1690    </dependency>
1691    <dependency>
1692       <groupId>javax.servlet.jsp</groupId>
1693       <artifactId>javax.servlet.jsp-api</artifactId>
1694       <version>2.3.3</version>
1695       <scope>provided</scope>
1696    </dependency>
1697    <dependency>
1698       <groupId>junit</groupId>
1699       <artifactId>junit</artifactId>
```

```
        <version>4.12</version>
        <scope>test</scope>
    </dependency>

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

7. FormValidationDemo Project > right-click > Properties > Project Facets > Select Java > Change Version 1.8
   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

      package com.example.vo;

      public class UserVO {
         private String name;
         private int age;
         private String userid;
         public String getName() {
            return name;
         }
         public void setName(String name) {
            this.name = name;
         }
         public int getAge() {
            return age;
         }
         public void setAge(int age) {
            this.age = age;
         }
         public String getUserid() {
            return userid;
         }
         public void setUserid(String userid) {
            this.userid = userid;
         }
         @Override
         public String toString() {
            return "UserVO [name=" + name + ", age=" + age + ", userid=" + userid + "]";
         }
      }


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

      package com.example.biz;
```

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

```
            page = "register";
        }
        return page;
    }
```

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

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>회원 가입 폼</title>
</head>
<body>
    <form action="/biz/register" method="post">
        Name : <input type="text" name="name" /><br />
        Age : <input type="number" name="age" /><br />
        ID : <input type="text" name="userid" /><br />
        <input type="submit" value="가입하기" />
    </form>
</body>
</html>
```

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

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="user" value="${userVO}" />
<!DOCTYPE html">
<html>
<head>
<meta charset="UTF-8">
<title>회원 가입 결과 창</title>
</head>
<body>
    <ul>
        <li>이름 : ${user.name}</li>
        <li>나이 : ${user.age}</li>
        <li>아이디 : ${user.userid}</li>
    </ul>
</body>
</html>
```

7)Test

http://localhost:8080/biz/register에서

이름, 나이, 아이디를 모두 입력하면 결과창으로 넘어오고

한 개라도 입력하지 않으면 다시 입력창으로 간다.

10. ValidataionUtils class를 이용한 검증

1)ValidatationUtils class는 validate() method를 좀 더 편리하게 사용할 수 있게 해줌.

2)UserValidator.java 수정

```
        /*String username = userVO.getName();
        if(username == null || username.trim().isEmpty()) {
           System.out.println("이름의 값이 빠졌습니다.");
           errors.rejectValue("name", "No Value");
        }*/

        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "name", "No Value");

11. @Valid와 @InitBinder 이용하기
  1)Spring Framework이 대신 검증해 줌
  2)mvnrepository에서 'hibernate validator'로 검색

     <dependency>
          <groupId>org.hibernate.validator</groupId>
          <artifactId>hibernate-validator</artifactId>
          <version>6.0.18.Final</version>
     </dependency>

  3)pom.xml에 넣고 Maven Clean > Maven Install
  4)HomeController.java 수정

     @RequestMapping(value = "/register", method=RequestMethod.POST)
     public String register(@ModelAttribute("userVO") @Valid UserVO userVO, BindingResult result) {
        String page = "register_ok";
        //UserValidator validator = new UserValidator();
        //validator.validate(userVO, result);
        if(result.hasErrors()) {
           page = "register";
        }

        return page;
     }

     @InitBinder
     protected void initBinder(WebDataBinder binder) {
        binder.setValidator(new UserValidator());
     }

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 > Check [Generate web.xml deployment descriptor] > Finish
4. Convert to Maven Project
  1)project right-click > Configure > Convert to Maven Project > Finish
  2)Project : /SpringWebDemo
```

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

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

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

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

-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>5.2.0.RELEASE</version>
  </dependency>
  <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.2.0.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>com.oracle</groupId>
    <artifactId>ojdbc6</artifactId>
    <version>11.2</version>
  </dependency>
  <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.2.0.RELEASE</version>
  </dependency>
</dependencies>
```

2)Maven Clean > Maven Install

7. 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

8. config folder에 beans.xml file 생성

1)Spring Perspective로 전환
2)config Folder > right-click > New > Spring Bean Configuration File
3)File name : beans.xml
4)생성시 beans,context, mvc 체크

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.2.xsd">


</beans>
```

9. 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:beans.xml</param-value>
</context-param>
```

10. 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>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

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

```
    <servlet-mapping>
      <servlet-name>springDispatcherServlet</servlet-name>
      <url-pattern>url</url-pattern>
    </servlet-mapping>

  2)아래의 코드로 변환
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:beans*.xml</param-value>
    </init-param>

    <servlet-mapping>
      <servlet-name>springDispatcherServlet</servlet-name>
      <url-pattern>*.do</url-pattern>
    </servlet-mapping>


11. UserVO class 생성
  1)src/com.example.vo package 생성
  2)src/com.example.vo.UserVO class 생성

    package com.example.vo;

    public class UserVO {
      private String userId;
      private String name;
      private String gender;
      private String city;
      public UserVO() {}
      public UserVO(String userId, String name, String gender, String city) {
        this.userId = userId;
        this.name = name;
        this.gender = gender;
        this.city = city;
      }
      public String getUserId() {
        return userId;
      }
      public void setUserId(String userId) {
        this.userId = userId;
      }
      public String getName() {
        return name;
      }
      public void setName(String name) {
        this.name = name;
      }
      public String getGender() {
        return gender;
      }
      public void setGender(String gender) {
        this.gender = gender;
```

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

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

```
2364
2365            @Override
2366            public UserVO read(String id) {
2367               String SQL = "SELECT * FROM users WHERE userid = ?";
2368               try {
2369                  UserVO user = jdbcTemplate.queryForObject(SQL, new Object[] { id }, new UserMapper());
2370                  return user;
2371               } catch (EmptyResultDataAccessException e) {
2372                  return null;
2373               }
2374            }
2375         }
2376
2377
2378 13. UserService 객체 생성
2379   1)src/com.example.service package 생성
2380   2)src/com.example.service.UserService interface
2381
2382      package com.example.service;
2383
2384      import java.util.List;
2385
2386      import com.example.vo.UserVO;
2387
2388      public interface UserService {
2389         void insertUser(UserVO user);
2390
2391         List<UserVO> getUserList();
2392
2393         void deleteUser(String id);
2394
2395         UserVO getUser(String id);
2396
2397         void updateUser(UserVO user);
2398      }
2399
2400   3)src/com.example.service.UserServiceImpl.java
2401
2402      package com.example.service;
2403
2404      import java.util.List;
2405
2406      import org.springframework.beans.factory.annotation.Autowired;
2407      import org.springframework.stereotype.Service;
2408
2409      import com.example.dao.UserDao;
2410      import com.example.vo.UserVO;
2411
2412      @Service("userService")
2413      public class UserServiceImpl implements UserService {
2414         @Autowired
2415         UserDao userDao;
```

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

15. config/dbinfo.properties file 생성

```
db.driverClass=oracle.jdbc.driver.OracleDriver
db.url=jdbc:oracle:thin:@localhost:1521:XE
db.username=hr
db.password=hr
```

16. beans.xml 수정

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

<context:property-placeholder location="classpath:dbinfo.properties" />
<bean id="dataSource" class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
   <property name="driverClass" value="${db.driverClass}" />
   <property name="url" value="${db.url}" />
   <property name="username" value="${db.username}" />
   <property name="password" value="${db.password}" />
</bean>
```

17. WebContent/index.jsp 생성

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

18. WebContent/userInfo.jsp 생성

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
   <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
   <c:set var="user" value="${user}"/>
   <!DOCTYPE html>
   <html>
   <head>
   <meta charset="UTF-8">
   <title>Insert title here</title>
   </head>
   <body>
      <h1>userInfo.jsp</h1>
      <h2>사용자 정보</h2>
      아이디 : ${user.userId} <br />
      이름 : ${user.name} <br />
      성별 : ${user.gender} <br />
      도시 : ${user.city} <br />
   </body>
   </html>
```

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 생선된 pom.xmlfile에 필요한 library 추가 > Maven Clean > Maven Install

```
<dependencies>
   <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.2.0.RELEASE</version>
   </dependency>
   <dependency>
       <groupId>org.springframework</groupId>
       <artifactId>spring-webmvc</artifactId>
       <version>5.2.0.RELEASE</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>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<!-- Map all requests to the DispatcherServlet for handling -->
<servlet-mapping>
  <servlet-name>springDispatcherServlet</servlet-name>
  <url-pattern>url</url-pattern>
</servlet-mapping>
```

2)아래의 코드로 변환

```
<init-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:beans.xml</param-value>
</init-param>

<servlet-mapping>
  <servlet-name>springDispatcherServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

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

```
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.4</version>
</dependency>
```

3)mvnrepository에서 'commons io'라고 검색하여 library 추가

```
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.6</version>
</dependency>
```

4)Maven Clean > Maven Install

10. Thumbnail Image Library 추가
1)mvnrepository에서 'imgscalr-lib'라고 검색하여 libary 추가

```
<dependency>
    <groupId>org.imgscalr</groupId>
    <artifactId>imgscalr-lib</artifactId>
    <version>4.2</version>
</dependency>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  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 mulipartResolver 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;
```

```
2721    import java.io.IOException;
2722    import java.util.Calendar;
2723
2724    import org.springframework.stereotype.Service;
2725    import org.springframework.web.multipart.MultipartFile;
2726
2727      @Service
2728      public class FileUploadService {
2729        // 리눅스 기준으로 file 경로를 작성 ( 루트 경로인 /으로 시작한다. )
2730        // 윈도우라면 workspace의 드라이브를 파악하여 JVM이 알아서 처리해준다.
2731        // 따라서 workspace가 C드라이브에 있다면 C드라이브에 upload 폴더를 생성해 놓아야 한다.
2732        private static final String SAVE_PATH = "/upload";
2733        private static final String PREFIX_URL = "/upload/";
2734
2735        public String restore(MultipartFile multipartFile) {
2736          String uri = null;
2737
2738          try {
2739            // file 정보
2740            String originFilename = multipartFile.getOriginalFilename();
2741            String extName = originFilename.substring(originFilename.lastIndexOf("."),
               originFilename.length());
2742            Long size = multipartFile.getSize();
2743
2744            // 서버에서 저장 할 file 이름
2745            String saveFileName = genSaveFileName(extName);
2746
2747            System.out.println("originFilename : " + originFilename);
2748            System.out.println("extensionName : " + extName);
2749            System.out.println("size : " + size);
2750            System.out.println("saveFileName : " + saveFileName);
2751
2752            writeFile(multipartFile, saveFileName);
2753            uri = PREFIX_URL + saveFileName;
2754          }
2755          catch (IOException e) {
2756            // 원래라면 RuntimeException 을 상속받은 예외가 처리되어야 하지만
2757            // 편의상 RuntimeException을 던진다.
2758            // throw new FileUploadException();
2759            throw new RuntimeException(e);
2760          }
2761          return uri;
2762        }
2763
2764
2765        // 현재 시간을 기준으로 file 이름 생성
2766        private String genSaveFileName(String extName) {
2767          String fileName = "";
2768
2769          Calendar calendar = Calendar.getInstance();
2770          fileName += calendar.get(Calendar.YEAR);
2771          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.
-즉 file을 root 경로인 / 아래의 upload folder에 저장하겠다는 의미인데, Windows에서는 JVM이 알아서 workspace가 존재하는 drive의 위치를 찾아서 drive를 root 경로로 하여 upload folder에 저장한다.
-예를들어 Eclipse workspace가 C drive에 있다면 C drive의 upload folder에 file이 저장될 것이다.
4)PREFIX_URL은 저장된 file을 JSP에서 불러오기 위한 경로를 의미한다.
5)MultipartFile 객체는 file의 정보를 담고 있다.
6)uri을 반환하는 이유는 view page에서 바로 image file을 보기 위함이다.
-만약 DB에서 image 경로를 저장 해야 한다면, 이와 같이 uri을 반환하면 좋을 것이다.
7)현재 시간을 기준으로 file 이름을 바꾼다.
-이렇게 하는 이유는, 여러 사용자가 올린 file의 이름이 같을 경우 덮어 씌어지는 문제가 발생하기 때문이다.
-따라서 file 이름이 중복될 수 있는 문제를 해결하기 위해 ms단위의 시스템 시간을 이용하여 file 이름을 변경한다.
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">
      <img src="${pageContext.request.contextPath }${uri }" style="width: 150px">
   </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

21. 문제점 및 해결

1)Upload Folder(C:/upload)를 보면 File이 Upload된 것을 확인할 수 있지만, 결과 화면을 보면 Image가 제대로 출력 되지 않을 것이다.

2)Image File을 right-click하여 경로를 보면 아마 다음과 같을 것이다.

3)http://localhost:8080/FileUploadDemo/upload/업로드한 파일

4)File을 저장할 때 [upload]라는 Folder에 저장을 했는데, File을 저장할 때의 Upload는 C Drive 내의 [upload] Folder이고,

5)위 URL에서 [upload]는 Application 상 경로에 있는 upload이므로 WEB-INF 폴더의 하위 folder로서의 upload를 의미한다.

6)즉 실제 File이 저장된 Server 상의 위치(물리 주소)와 Application에서 보여주고자 하는 File 경로(가상 주소)가 일치하지 않은 것이다.

7)따라서 실제 File이 저장되어 있는 위치와 Application 상의 위치를 일치시키는 작업이 필요하다.

8)beans.xml에 물리 주소와 가상 주소를 mapping 해주는 code를 추가하도록 해야한다.

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

9)이제 정상적으로 result.jsp에서 image가 출력될 것이다.

22. Multiple File Upload

1)이번에는 여러 개의 File을 Upload 할 수 있는 Multiple Upload를 알아보자.

2)수정할 부분은 <input> tag와 Controller에서 MultipartFile 객체를 받는 Parameter 부분 두 곳인데, 필요한 부분만 보자.

3)form.jsp

```
<input type="file" name="files" multiple>
```

4)<input> 태그에서는 multiple 속성만 추가하면 된다.

5)"File선택"을 클릭하면 ctrl 키를 눌러서 여러 개의 File을 선택할 수 있다.

6)FileUploadController

```
@RequestMapping( "/upload" )
public String upload(@RequestParam String email,
    @RequestParam(required=false) List<MultipartFile> files, Model model) {

      ...

}
```

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